UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA

INGENIRÍA TÉCNICA INDUSTRIAL

(ELECTRÓNICA INDUSTRIAL)

DESARROLLO DE MÓDULOS PARA FPGA DE PLATAFORMA DE CONTROL DE CONVERTIDORES

Alumno: Juan Pablo Ruiz-Garrido Seoane
Tutor por la universidad: Mario García Valderas
Tutor por la empresa: Fernando Casado Ortiz

Dedico el presente proyecto a Fernando Casado, sin cuya ayuda no hubiera sido posible completar el proyecto.

También quiero mencionar el ánimo y apoyo prestado por mis padres, Mª. Inés y José Felix.

RESUMEN

En este proyecto se han diseñado dos módulos en VHDL para una plataforma de control de convertidores estáticos ferroviarios.

El primer módulo almacena las muestras de señales analógicas y digitales provenientes de tarjetas de adquisición exteriores (tarjetas I/O). Para ello se ha empleado una memoria SDRAM DDR funcionando a modo de LIFO, en la que los datos se clasificarán en función el tiempo en el que han sido adquiridas y del lugar de procedencia. Cuando se produzca una avería en el convertidor, se podrá solicitar una descarga de las muestras de la memoria DDR, con el fin de facilitar la búsqueda del origen de la avería.

El segundo módulo implementa el algoritmo de la DFT (transformada discreta de Fourier). Se proporciona como salida el valor eficaz de una componente armónica a partir del módulo de la DFT. El módulo es configurable para un número programable de señales y/o componentes armónicas.

Palabras clave: FPGA, DFT, VHDL, DDR, SDRAM, I/O, valor eficaz, traza de averías, plataforma de control.

ABSTRACT

In this Project two VHDL modules were designed for a control platform applied to railway static converters.

The first module stores analog and digital signals coming from external acquisition boards (I/O boards). For that purpose a SDRAM DDR operating as LIFO was used, in which all incoming data will be classified as a function of sampling time and external source. When a failure in the converter occurs, it will be possible to request a download of the samples stored in the DDR, in order to facilitate troubleshooting.

The second module implements the DFT algorithm (Discrete Fourier Transform). It provides as output the RMS of a harmonic component from the DFT module. The module is configurable for a programmable number of signals and/or harmonic components.

Keywords: FPGA, DFT, VHDL, DDR, SDRAM, I/O, RMS, failures trace, control platform.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

LISTADO DE ACRÓNIMOS

ADC: Analog to digital converter (conversor analógico/digital).

DAC: Digital to analog converter (conversor digital/analógico).

DCM: Digital clock manager.

DDR: Double data rate (doble frecuencia de transferencia de datos).

DFT: Discrete Fourier transform (transformada discreta de Fourier).

FIFO: First input first output.

FPGA: Field programmable gate array.

GUI: Graphical user interface (interfaz gráfica de usuario).

IGBT: Insulated Gate Bipolar Transistor.

LIFO: Last input first output.

MIG: Memory interface generator (generador de interfaces de memoria).

MUX: Multiplexer (multiplexor).

MVB: Multifunction Vehicle Bus.

RAM: Random Access Memory.

RTL: Register transfer logic (lógica de transferencia de registro).

SDC: Synopsys design constraints (Restricciones de diseño de Synopsys).

SDRAM: Synchronous Dynamic Random Access Memory.

SPI: Serial Peripheral Interface (interfaz periférico serie).

THD: Total harmonic distorsion (distorsión armónica total).

UCF: User constraints file (fichero de restricciones del usuario).

VHDL: VHSIC hardware description language (lenguaje de descripción hardware).

VHSIC: Very high speed integrated circuit (Circuito integrado de alta velocidad).

1 Introducción

Dentro de los sistemas de control de convertidores estáticos ferroviarios, es cada vez más común la inclusión de FPGAs que implementan funciones de cálculo y procesamiento de datos que son empleadas en tareas de regulación, protección y comunicaciones. Tradicionalmente estas tareas de control eran implementadas en microcontroladores. Las ventajas de las FPGAs frente a los microcontroladores son las siguientes:

- Aumento de las prestaciones.
- Aumento de la velocidad.
- Arquitectura optimizada para el cálculo.
- Reutilización de los diseños.

A pesar de ello, las FPGAs presentan una serie de inconvenientes:

- Un precio superior a los microcontroladores.
- Unos tiempos de desarrollo elevados (principalmente en el primer diseño) que repercuten directamente en el coste del diseño.

Los convertidores del mundo ferroviario presentan unos precios elevados. Los componentes que más influyen en el coste son los elementos magnéticos (bobinas y transformadores). Analizando el impacto de la FPGA sobre el coste total, se llega a la conclusión de que va a tener un impacto despreciable sobre el coste del convertidor.

1.1 Descripción general de una plataforma de control de convertidores

Para el desarrollo de este proyecto se ha hecho uso de la siguiente plataforma de control de convertidores. Tal como muestra la Ilustración 1.

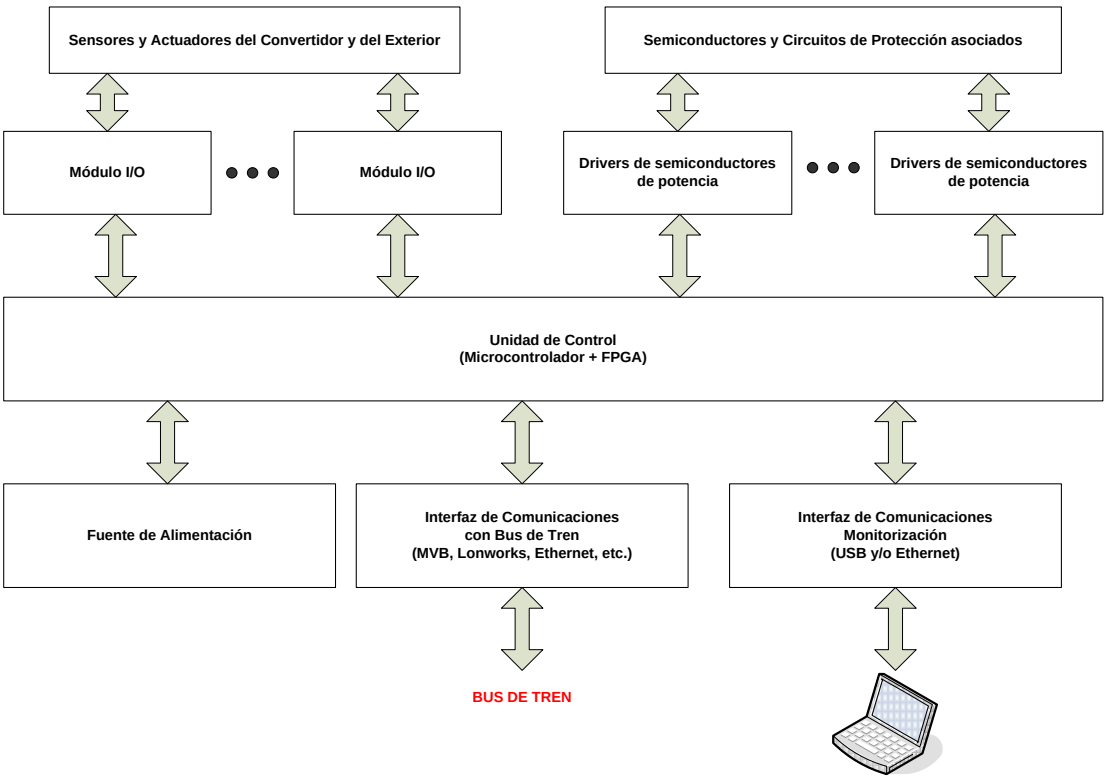

Ilustración 1: Diagrama de bloques de la plataforma de control de convertidores

A continuación se describen los componentes de la plataforma de control:

1. **Unidad de Control.** Centraliza las tareas de regulación y protección de los Sistemas Electrónicos de Potencia del convertidor, así como la gestión de las comunicaciones con el Bus de Tren y el sistema de monitorización basado en PC. Está gobernada por un microcontrolador y una FPGA. Estos componentes dotan a la unidad de control de una capacidad de cálculo que permite por una parte emplear algoritmos de control sofisticados y por otra la implementación de funciones que ayudan a la detección de averías y funcionamientos anómalos del convertidor (p.e. traza de eventos y transitorios con una granularidad temporal fina o el análisis frecuencial de formas de onda empleando la DFT).

2. **La fuente de alimentación** genera las diferentes tensiones de alimentación de la plataforma a partir de la tensión de batería del tren.

3. **El Interfaz de Comunicaciones con el Bus de Tren** contiene el controlador de comunicaciones con el Bus de Tren (MVB, LonWorks, Ethernet, etc.).

4. **El Interfaz de Comunicaciones de Monitorización** contiene el controlador de comunicaciones con el Sistema de Monitorización basado en PC (USB, Ethernet, etc.).

5. **Módulo I/O.** Se encarga de la gestión de las señales analógicas y digitales intercambiadas entre la Unidad de Control y los sensores y actuadores del convertidor o del exterior. Las entradas analógicas y digitales se adaptan y se filtran con circuitos personalizados a cada tipo de convertidor. Dichas entradas

son muestreadas y enviadas a la Unidad de Control a través de un Bus serie con la ayuda de conversores A/D y registros de desplazamiento.

6. **Drivers de Semiconductores de Potencia.** Los comandos de activación de los semiconductores, enviados por la Unidad de Control se adaptan y se aíslan galvánicamente antes de ser aplicados a las entradas de activación de los semiconductores.

1.2 Objetivos del proyecto

El presente proyecto pretende cubrir dos funciones fundamentales dentro de la plataforma de control del convertidor:

1. **Diseño de una traza de averías (módulo de muestreo y gestión de datos en memoria).** Se va a diseñar un módulo capaz de almacenar las muestras proporcionadas por los módulos I/O. Para ello se empleará una memoria SDRAM DDR funcionando a modo de LIFO, en la que se guardarán las muestras y además, se clasificarán en función el tiempo en el que han sido adquiridas y del lugar de procedencia. El tamaño de la DDR es configurable en función del número de módulos I/O. Cuando se produzca una avería en el convertidor, se podrá solicitar una descarga de las muestras de la memoria DDR, con el fin de facilitar la búsqueda del origen de la avería. La Ilustración 2 muestra el flujo de datos para la traza de averías:

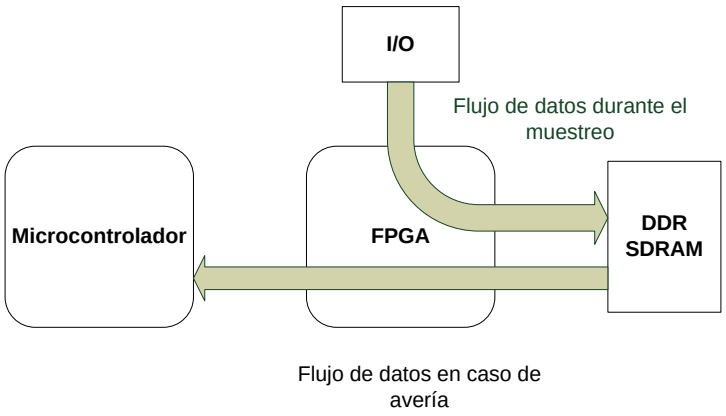

Ilustración 2: Flujo de datos en la traza de averías

El diseño del módulo de muestreo y gestión de datos en memoria se ha hecho en dos etapas. En una primera etapa se diseño un módulo que cumplía con los requisitos de velocidad a costa de un gran consuno de área. En una segunda etapa se optimizó el diseño de cara a aumentar la relación área/velocidad.

2. **Algoritmo de la DFT (módulo de la DFT).** La DFT permite conocer el valor eficaz de cualquier armónico presente en una señal. La utilización del algoritmo en éste caso, permite desarrollar las siguientes funcionalidades:

- **Detección de averías en los IGBTs del inversor**. La componente fundamental de la corriente de entrada al inversor aporta información sobre la integridad de los IGBTs del inversor. Cuando el valor eficaz de la componente fundamental de la corriente supere un valor umbral se interpretará como una avería de al menos uno de los IGBTs (IGBT abierto).

- **Cálculo de la Distorsión armónica total (THD)**: para realizar dicho cálculo es preciso conocer el valor eficaz de la componente fundamental. La DFT permite obtener el valor eficaz de la componente fundamental.

Para la verificación de la traza de averías y la DFT se ha empleado la placa Spartan-3A/3AN FPGA Starter Kit de Xilinx. La ventaja de la utilización de ésta placa es una mayor disposición de recursos hardware en la etapa de validación de los diseños (leds, switchs, ADCs, DACs, etc). Dicha placa contiene una memoria SDRAM DDR2 (no una DDR). Por lo que se requiere una traza de averías que sea compatible con las memorias DDR y DDR2 cambiando únicamente el controlador de la memoria generado con la herramienta MIG.

Tanto la traza de averías como la DFT se implementarán sobre la tarjeta "Spartan-3E FPGA Industrial Micromodule" de la compañía Trenz Electronic.

1.3 Estructura del proyecto

En el capítulo 2 se va a describir el módulo de muestreo y gestión de datos en memoria. Este módulo implementa la función de traza de averías. Se comenzará proporcionando una visión global del módulo y la forma en la que interactúan sus componentes fundamentales. Una vez que se otorgue una visión global del conjunto del sistema, se describirá con detalle cada componente por separado. Finalmente se validará el módulo mediante simulación e implementación en una FPGA.

En el capítulo 3 se describe el módulo de la DFT. Inicialmente se expondrá el funcionamiento del algoritmo, sus parámetros fundamentales y su utilidad dentro de la plataforma de control del convertidor. A continuación se estudiará y dimensionará el algoritmo con el software Matlab. Una vez se halla dimensionado el algoritmo, se va a describir como se ha digitalizado el algoritmo. Como último paso se validará el módulo mediante implementación y validación en una FPGA.

El capítulo 4 se corresponde con el presupuesto. El presupuesto se ha subdividido en dos grupos, presupuesto de material y presupuesto de personal.

En el capítulo 5 se exponen las conclusiones y las líneas de investigación futuras.

El capítulo 6 es la bibliografía, donde se pueden consultar todas las referencias que se han utilizado para el desarrollo del proyecto.

2 Módulo de muestreo y gestión de datos en memoria

En este capítulo se van a tratar las diferentes arquitecturas que componen el módulo de muestreo y gestión de datos en memoria (traza de averías). El esquema general de la traza de averías de la plataforma de control es el siguiente:

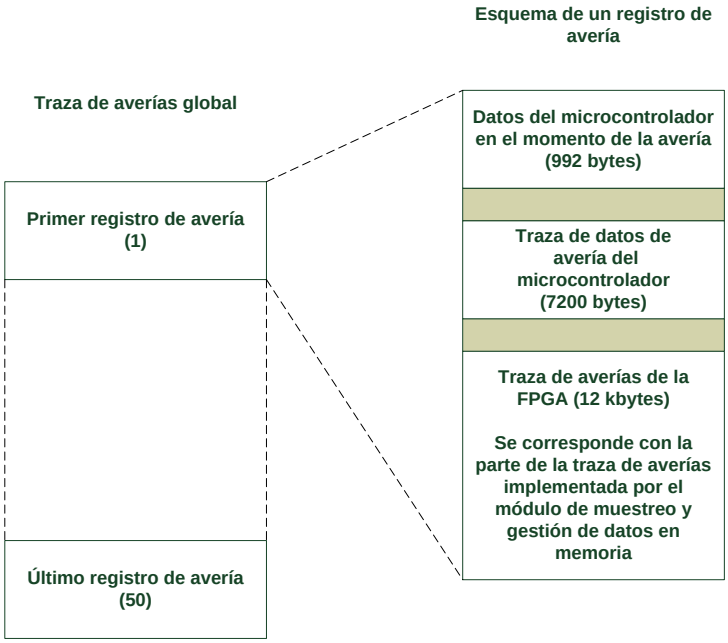

Ilustración 3: traza de averías de la plataforma de control

En la Ilustración 3 se puede apreciar la parte de la traza de averías que se pretende implementar con el módulo de muestreo y gestión de datos en memoria.

En la siguiente imagen se puede observar el diagrama de bloques del sistema:

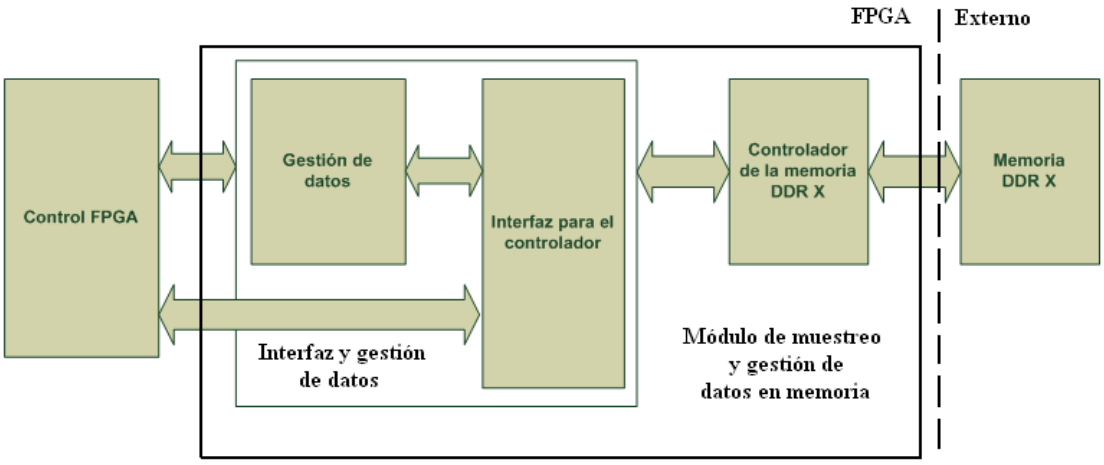

Ilustración 4: Diagrama de bloques del sistema.

El objetivo es diseñar una traza de averías del convertidor, tomando las muestras proporcionadas por las tarjetas I/O. A tal fin se emplea una memoria SDRAM funcionando a modo de LIFO, esto es, las primeras muestras en escribirse son las últimas en leerse. Es importante implementar una clasificación de las muestras en función del tiempo en el que se han adquirido y del lugar de procedencia.

Como se observa en el diagrama de bloques, el módulo de muestreo y gestión de datos en memoria consta de los siguientes elementos:

- Memoria SDRAM DDR X. X hace referencia a la memoria DDR2 Y DDR.
- Controlador de la memoria DDR X. Reduce la complejidad del control de la memoria y además, implementa el protocolo DDR para operaciones de lectura/escritura en memoria. Dicho protocolo permite la transferencia de dos datos simultáneamente en el mismo ciclo de reloj.
- Interfaz para el controlador y gestión de datos. El interfaz permite realizar operaciones de escritura/lectura, implementando sobre la memoria una funcionalidad de LIFO. La gestión de datos clasifica las muestras proporcionadas por las tarjetas I/O y escribe éstas en el interfaz. Se han realizado dos diseños para el interfaz, en uno de ellos (acceso continuo), la gestión de datos está implementada en el propio interfaz. En el otro (interfaz de acceso esporádico), la gestión de datos constituye un bloque aparte.

El sistema tiene cinco estados diferentes de funcionamiento: escritura, lectura, inicialización de la memoria, test de la memoria y reposo.

- **Escritura**: el sistema escribe en memoria datos de 16 bits. El sistema debe ser capaz de clasificar los datos en función del tiempo de adquisición y del lugar de procedencia (aumentando para ello el ancho del dato a 32 bits) y escribirlos en memoria, sin que se pierda ninguna muestra en el proceso.
- **Lectura**: El sistema lee los datos de la memoria DDR y los pone a disposición del control de la FPGA.
- **Inicialización de la memoria**: Se inicializan todas las posiciones de la pila utilizada en memoria.
- **Test**: Se verifica la funcionalidad de todas las celdas de memoria utilizadas en la pila.
- **Reposo**: el sistema permanece en un estado de reposo a la espera de nuevas órdenes por parte del control de la FPGA.

2.1 Diseño del Controlador de la memoria DDR y DDR2 con la herramienta MIG de Xilinx

2.1.1 Descripción general de la herramienta MIG (Memory Interface Generator de Xilinx)

El Memory Interface Generator (MIG) es una herramienta del "Core Generator" dentro del software ISE de Xilinx. La función del MIG es generar controladores de memoria para las siguientes memorias y FPGAs: DDRII SRAM, DDR SDRAM, DDR2 SDRAM, QDRII SRAM y RLDRAM II para la Virtex®-4, DDR SDRAM, DDR2 SDRAM, QDRII SRAM y DDRII SRAM para la Virtex-5. También las memorias DDR y DDR2 SDRAM para las FPGAs Spartan®-3, Spartan-3A, Spartan-3E y Spartan-3A.

Para generar un controlador de memoria, el MIG requiere unas entradas, como pueden ser tipo de memoria, familia de la FPGA, dispositivos de la FPGA, frecuencia, ancho de dato, modo en el que el controlador registra los valores, etc. La introducción de estos valores se realiza mediante un interfaz gráfico, "Graphical User Interface" (GUI).

Como salidas el MIG proporciona: archivos RTL, SDC, UCF, script y diversos documentos. Los archivos script son usados para: simulación, síntesis, mapeo y rutado para la configuración seleccionada. Los archivos de diseño RTL pueden ser generados en VHDL o en Verilog, dependiendo del tipo de memoria y FPGA seleccionadas en el diseño.

El MIG puede realizar las siguientes tareas:

1. Crear un diseño.
2. Crear un diseño para una placa específica de Xilinx.
3. Verificar y actualizar el UCF de un diseño.

Finalmente, los diseños generados con el MIG pueden ser simulados mediante los ficheros proporcionados en la carpeta SIM. Esta carpeta proporciona todo lo necesario para realizar una correcta simulación, incluyendo tanto un banco de pruebas, así como otros módulos necesarios. El banco de pruebas genera el reloj, el reset y las señales de entrada del sistema.

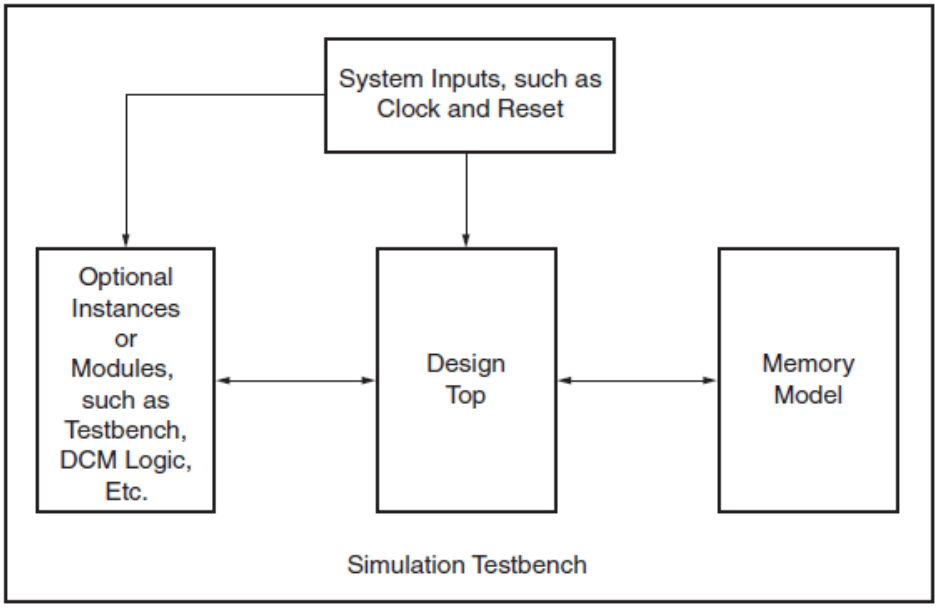

Ilustración 5: Diagrama de bloques del entorno de simulación

Los elementos proporcionados para la simulación son:

1. **Memory Model:** Es el modelo para la simulación. Se proporciona siempre en Verilog.

2. **Optional Instances:** En un diseño en el que se incluye el banco de pruebas proporcionado por el MIG, éste incluye todos los componentes necesarios para la simulación del diseño. Si no se incluye el banco de pruebas del MIG, ésta parte corresponderá a la lógica realizada por el usuario.

3. **Design Top:** En el diseño de ejemplo esta parte conecta con los resets, relojes, señales de estado y las señales del interfaz de la memoria (todo esto proporcionado por el MIG en la carpeta SIM). En el diseño del usuario, ésta parte conecta con las señales del interfaz, los resets, los relojes y las señales de estado del usuario.

2.1.2 Implementación del controlador DDR2 con el MIG

Es importante tener en cuenta a la hora de realizar el diseño con el MIG no solo la FPGA empleada en el diseño sino también la conexión física entre la memoria DDR2 y la FPGA. En este caso la memoria DDR2 a controlar monta sobre el kit Spartan-3A/3AN FPGA Starter Kit de Xilinx. La FPGA es una Spartan3-700AN. Más adelante se verá la importancia del conexionado físico de la memoria para la creación del controlador.

La memoria incluida en el kit de Xilinx es: DDR2 SDRAM (MT47H32M16). Con una capacidad de 512 Mbit y un formato de 32M x 16 bits. El fabricante es Micron Technology.

El primer paso, es crear un nuevo proyecto. En nuestro caso seleccionamos las siguientes características:

Property Name	Value
Top-Level Source Type	HDL
Product Category	All
Family	Spartan3A and Spartan3AN
Device	XC3S700AN
Package	FGG484
Speed	-4
Synthesis Tool	XST (VHDL/Verilog)

Ilustración 6: Características del proyecto para el controlador DDR2

El siguiente paso es añadir un nuevo fichero al proyecto de tipo IPCORE. En este caso se seleccionará el MIG versión 3.6, iniciándose la herramienta gráfica GUI (Ilustración 7). Tras seleccionar el botón de Next, la herramienta pregunta sobre la actividad que se quiere realizar, que en este caso es la creación de un diseño (create design).

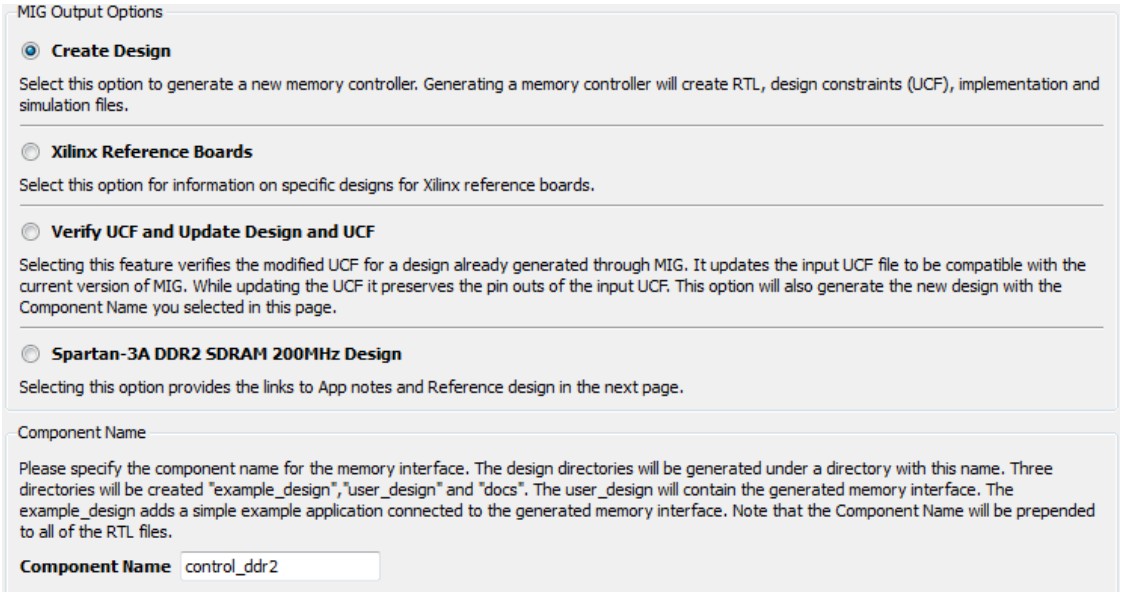

Ilustración 7: Selección de tarea a realizar con el MIG.

Etapas en la creación de un diseño:

1) **Selección de FPGAs con pines compatibles:** La herramienta ofrece la posibilidad de generar pines compatibles para FPGAs con un mismo grado de velocidad y un mismo encapsulado. Los pines se generan en el UCF del diseño. En este diseño no se ha hecho uso de esta opción.

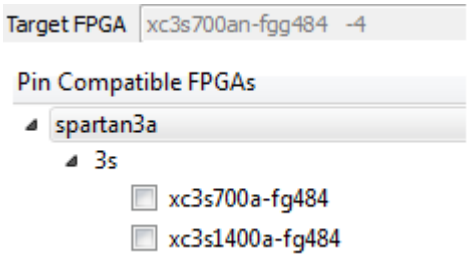

Ilustración 8: Pin Compatible FPGAs.

2) **Selección de la memoria (Memory selection):** En esta parte del diseño se selecciona el tipo de memoria a controlar.

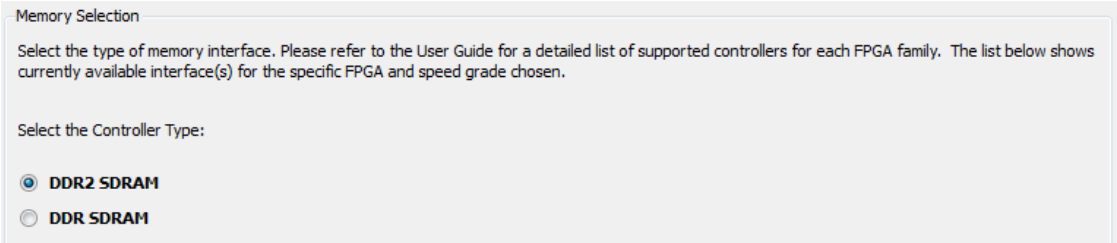

Ilustración 9: selección de la memoria en el MIG

3) **Opciones del Controlador (Controller options):** Se selecciona la frecuencia (125 MHz), etapas de escritura, características de la memoria y la opción de incluir una máscara de datos. En Memory Part, H32M16 hace referencia al tamaño de la memoria, 32MWORDS.

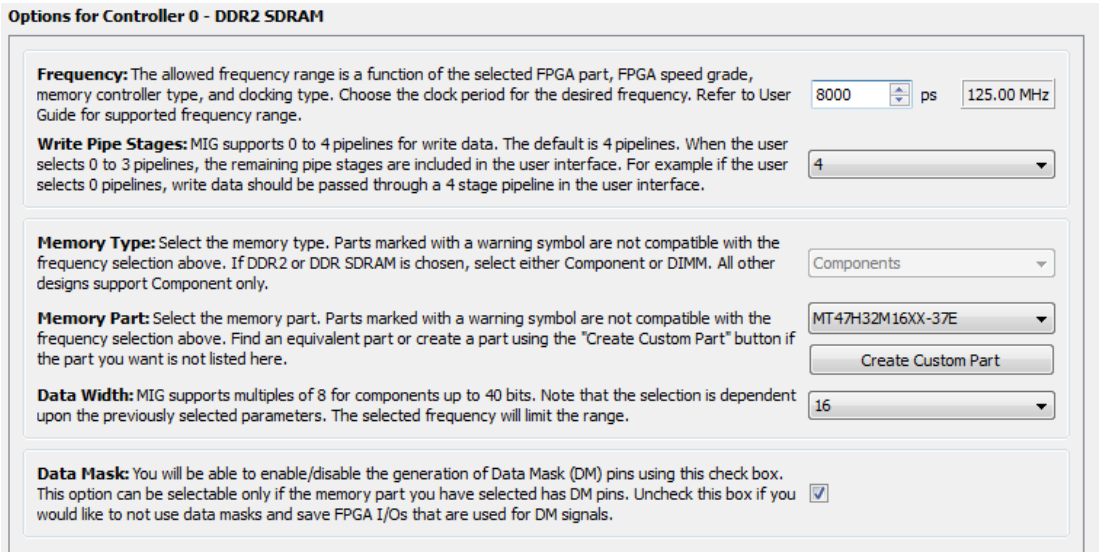

Ilustración 10: Opciones del controlador para la DDR2.

4) **Opciones de la memoria (Memory Options):** Se seleccionan parámetros de control. Burst Lenght especifica el número de columnas a las que se accede con una escritura o lectura, en este caso es 4, por lo que cada vez que se escribe o lee una dirección se utilizan 4*16 bits. Burst type especifica el orden de acceso. RTT se emplea en la selección de resistencias terminales internas para las

señales listadas en la Ilustración 11. DQSENABLE, permite elegir un modo de transmisión diferencial o de una sola terminación.

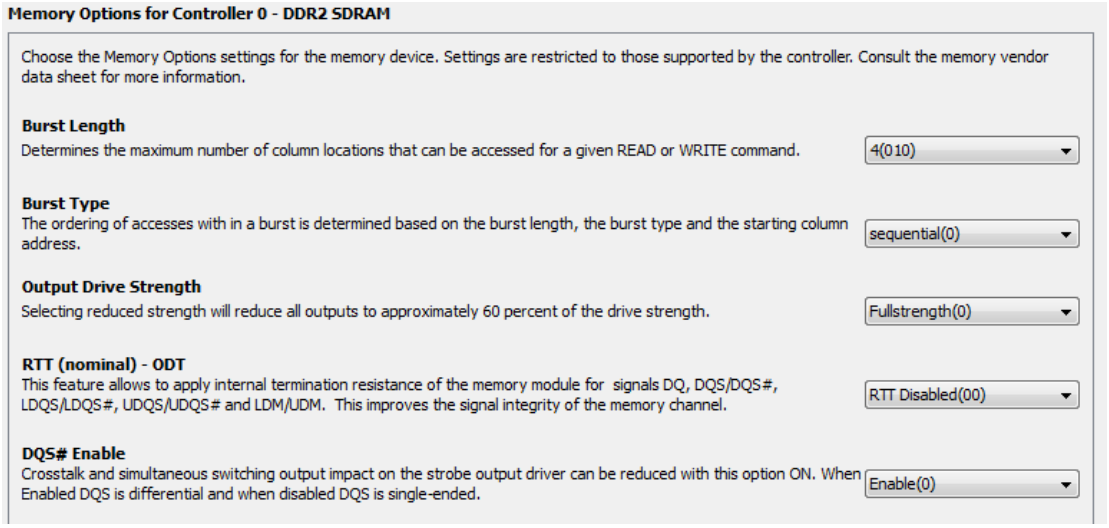

Ilustración 11: Memory options DDR2.

5) **Opciones de la FPGA** (FPGA Options): se selecciona la clase de control, la posibilidad de inclusión de un DCM en el diseño (en caso de generar un DCM con MIG, se pide el tipo de reloj de entrada, diferencial o de una sola terminación) y la posibilidad de usar señales para ser monitorizadas en el ChipScope referentes a las señales de calibración.

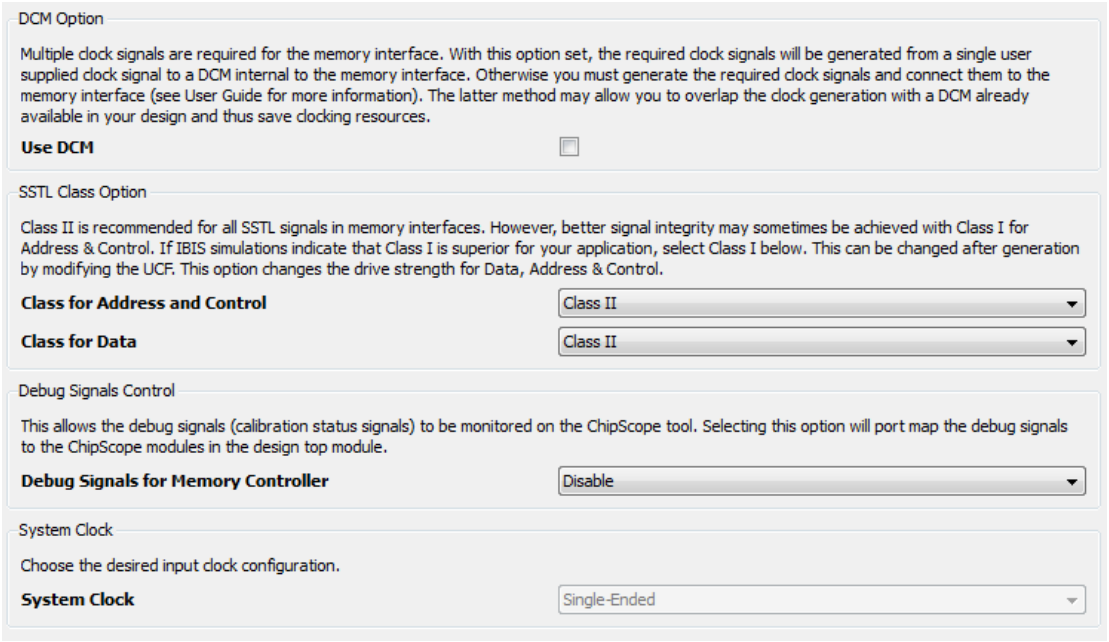

Ilustración 12: FPGA Options DDR2.

6) **Pines reservados (Reserved Pins):** En este proyecto no se ha hecho uso de esta herramienta. Los pines utilizados en el diseño se han incluido en el UCF del proyecto. Esta herramienta proporciona la posibilidad de especificar los pines de la FPGA que no se usan en la creación del controlador de la memoria.

7) **Selección del emplazamiento (Bank Selection):** se selecciona el emplazamiento del control en la FPGA.

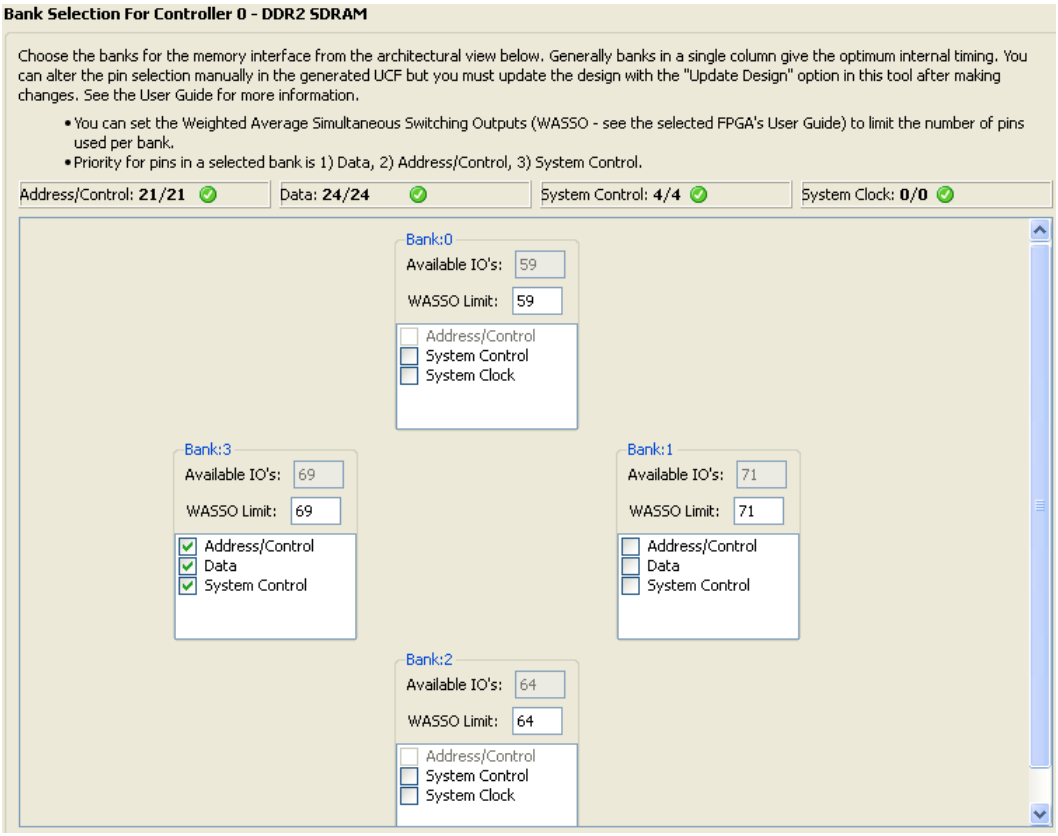

Ilustración 13: Bank Selection DDR2.

"Bank Selection" permite seleccionar en que zona de la FPGA va a estar físicamente el controlador. Es muy importante que se seleccionen bien los bancos de la FPGA, teniendo implicaciones directas en las etapas de calibración de la memoria y el retardo en la propagación de las señales. Una mala selección puede suponer una pérdida de datos tanto en la lectura como en la escritura de la memoria.

A continuación se muestra en que banco de la FPGA están emplazados los pines conectados físicamente a la memoria DDR2 justificando los valores de Bank Selection escogidos en la Ilustración 14.

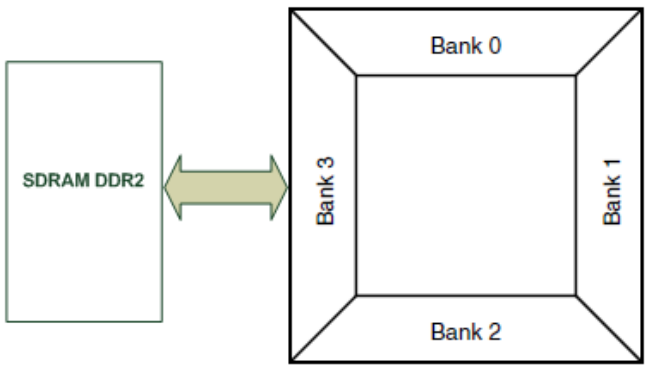

Ilustración 14: Conexionado DDR2 FPGA

Se puede observar como el banco 3 de la FPGA es el único que interactúa con la memoria DDR2, por tanto, debe generarse la mayor parte del controlador en la lógica cercana al banco 3 de la FPGA.

Seguidamente se obtiene el sumario del diseño generado, en el que se debe comprobar la correcta selección de parámetros en el GUI (se incluye en el CD).

Una vez se está de acuerdo con el sumario del diseño, se debe aceptar la licencia del modelo de simulación. Tras aceptar la licencia y leer las recomendaciones proporcionadas por el MIG se puede generar el controlador de la memoria DDR2.

El MIG proporciona todos los ficheros del controlador en la carpeta RTL (es la parte que finalmente se sintetizará). En la carpeta SIM se proporcionan todos los ficheros de simulación, como son el modelo de la memoria, el banco de pruebas, generadores de datos, de direcciones y de retardos. Además proporciona un UCF compatible con la FPGA utilizada en el diseño. En este UCF, se definen los pines de la DDR2, las especificaciones de tiempo, y el emplazamiento de ciertos componentes del controlador en la FPGA. La utilización del UCF es opcional, aunque es muy recomendable para evitar problemas en el calibrado de la memoria.

Finalmente se obtiene el controlador de la memoria DDR2 SDRAM.

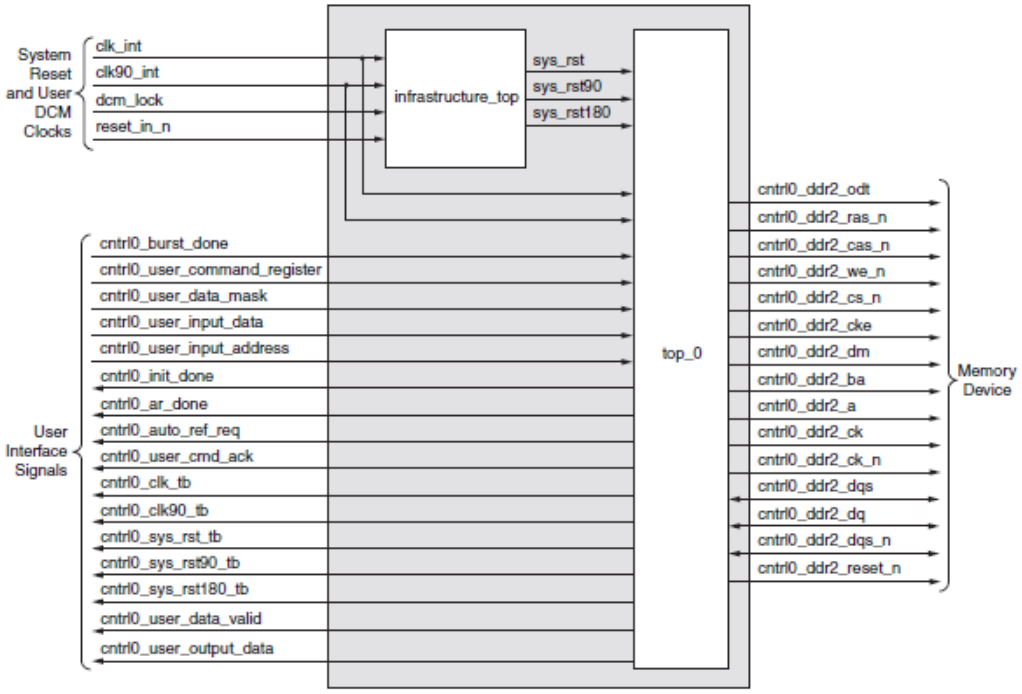

Ilustración 15: Diagrama de bloques DDR2.

El controlador de la memoria solamente interactúa con el interfaz para el controlador y con la memoria DDR, a excepción de la señal de reset_in, dcm_lock (conectadas a la señal de reset global del sistema) y los dos relojes de entrada del controlador de 125MHz, con uno de ellos desfasado un cuarto de ciclo respecto del otro.

Modificaciones en el UCF generado por el MIG:

En el fichero UCF, proporcionado automáticamente por el MIG al finalizar el diseño del controlador, se especifican una serie de características relativas a emplazamiento de las etapas de calibración, zona de la FPGA en la que se debe emplazar el controlador, máximos retardos permitidos, etapas de selección de datos, asignación de pines, etc. No se va a entrar en detalle sobre el funcionamiento de algunas de estas etapas y características, pero si es necesaria la corrección y/o modificación de ciertos apartados del fichero UCF.

Modificación del emplazamiento del controlador:

En el UCF generado por el MIG, el usuario puede elegir el banco de la FPGA en el que se va a emplazar el bus de direcciones, el bus de datos, el sistema de control, etc. Pero en ningún caso puede elegir directamente los pines de la FPGA conectados a la memoria. Esto es, el MIG selecciona los pines de la FPGA entre el banco seleccionado por el usuario y entre los que no se han especificado como pines prohibidos. Esto no tendría mayor inconveniente que cambiar en el UCF los pines proporcionados por el MIG por los que realmente emplea el kit de Xilinx. El problema viene de que al modificar los pines se deben modificar también las etapas de retardo asociadas al pin, los circuitos de captura, de calibrado, además de cumplir una seria de reglas relativas al emplazamiento (Chapter 14, "Debugging MIG DDR2 Designs", documento UG086).

Al estar diseñandose un controlador para una placa específica de Xilinx, se puede hacer uso de la opción "Xilinx Reference Boards" de la Ilustración 7. Xilinx proporciona UCFs completos para la implementación de controladores de memoria DDRX en sus placas específicas. Por lo tanto, en este caso se hará uso del UCF proporcionado por Xilinx para la placa Spartan-3A/3AN FPGA Starter Kit.

Si se desea ampliar información sobre cualquier tema tratado en este apartado relativo al controlador de la DDR2, se recomienda consultar el documento UG086 de Xilinx (pags 313-352).

El documento "Spartan-3A/3AFPGA Starter Kit Board User Guide" UG334 de Xilinx contiene toda la información relativa a la placa sobre la que se ha realizado el presente diseño.

El funcionamiento de la memoria DDR2 se puede consultar en la hoja de características "512 Mb DDR2 SDRAM". Las características y el fabricante de la memoria se adjuntan al inicio del presente apartado.

2.1.3 Implementación del controlador DDR con el MIG

El diseño del controlador DDR se ha realizado sobre una FPGA Spartan3-1600E. En este caso la placa sobre la que va montada la FPGA es "Spartan-3E FPGA Industrial Micromodule" de la compañía Trenz Electronic.

La memoria incluida en el kit de Xilinx es: DDR SDRAM (MT46V32M16). Con una capacidad de 512 Mbit y un formato de 32M x 16 bits. El fabricante es Micron Technology.

El procedimiento es completamente análogo al descrito en el apartado anterior. Primeramente se exponen las características de la FPGA sobre la que se va a trabajar:

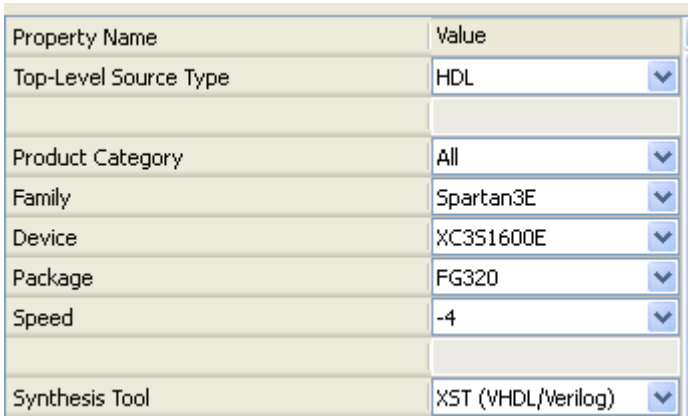

Ilustración 16: Características del proyecto para el controlador de la DDR

Se vuelve a acceder al GUI, indicándose la creación de un nuevo diseño. El proceso del diseño consta del mismo procedimiento, la diferencia radica, en las opciones que permite escoger el GUI.

1) **Selección de FPGAs con pines compatibles:** No se selecciona ninguna otra FPGA compatible con el diseño.

2) **Selección de la memoria (**Memory Selection): Se selecciona la memoria SDRAM DDR (la única opción disponible).

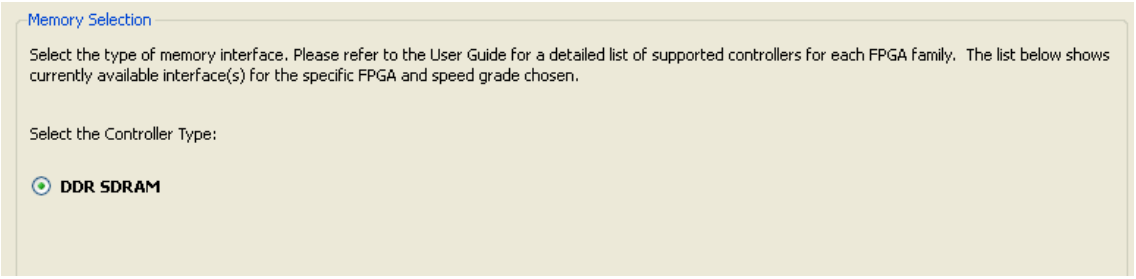

Ilustración 17: Selección del tipo de memoria

3) **Opciones del Controlador (Controller Options):** Se selecciona la frecuencia de funcionamiento (80 MHz), las etapas de escritura, el ancho de las datos de la memoria (16 bits) y la opción de incluir una máscara de datos (en este caso no se utiliza). V32M16 hace referencia a 32 Mwords.

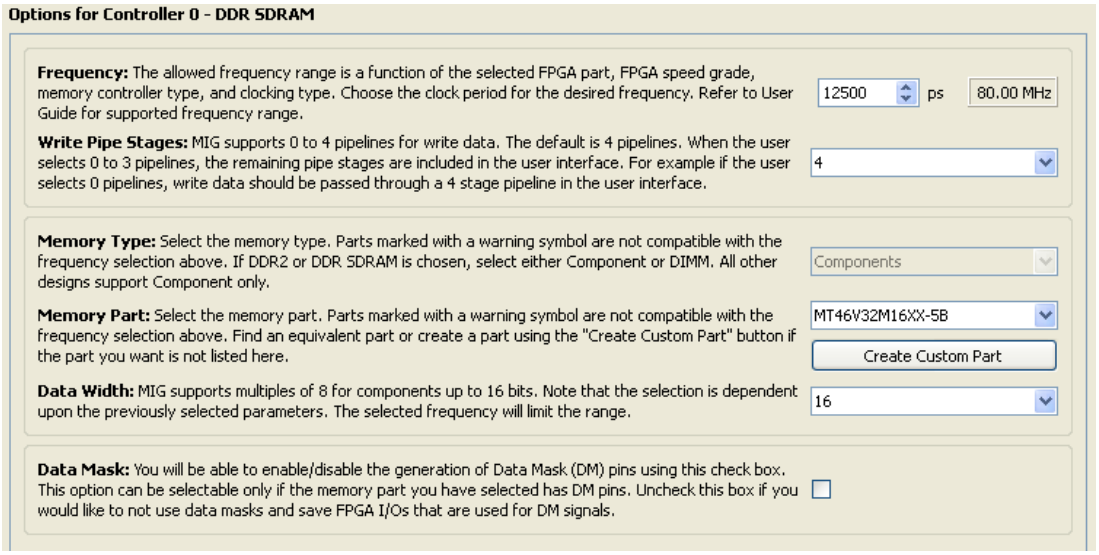

Ilustración 18: Opciones del controlador DDR

4) **Opciones de la memoria (Memory Options):** La descripción es análoga a la del apartado 2.1.2.

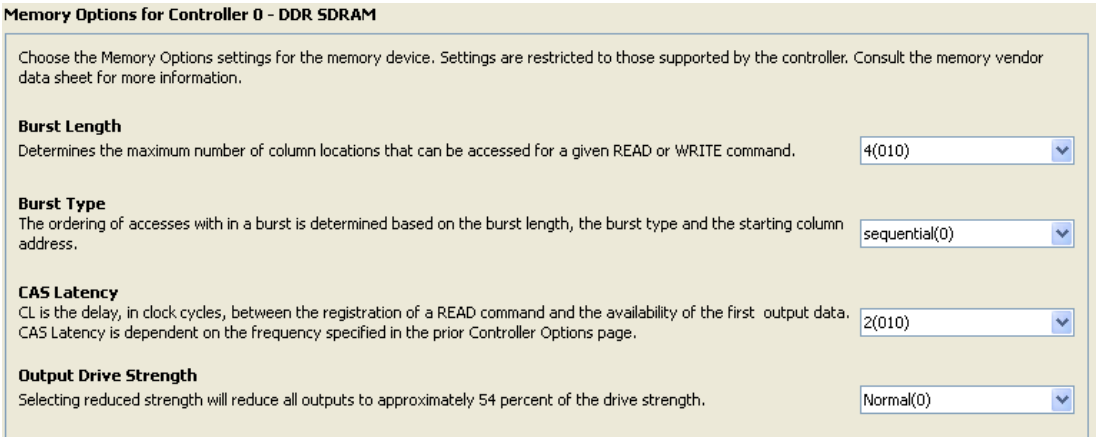

Ilustración 19: Memory Options DDR.

5) **Opciones de la FPGA (FPGA Options):** Se rechaza la inclusión del DCM en el diseño y se selecciona el tipo de control empleado. Tampoco se va a trabajar con el ChipScope, por lo que se rechaza la opción de Debug Signals Control.

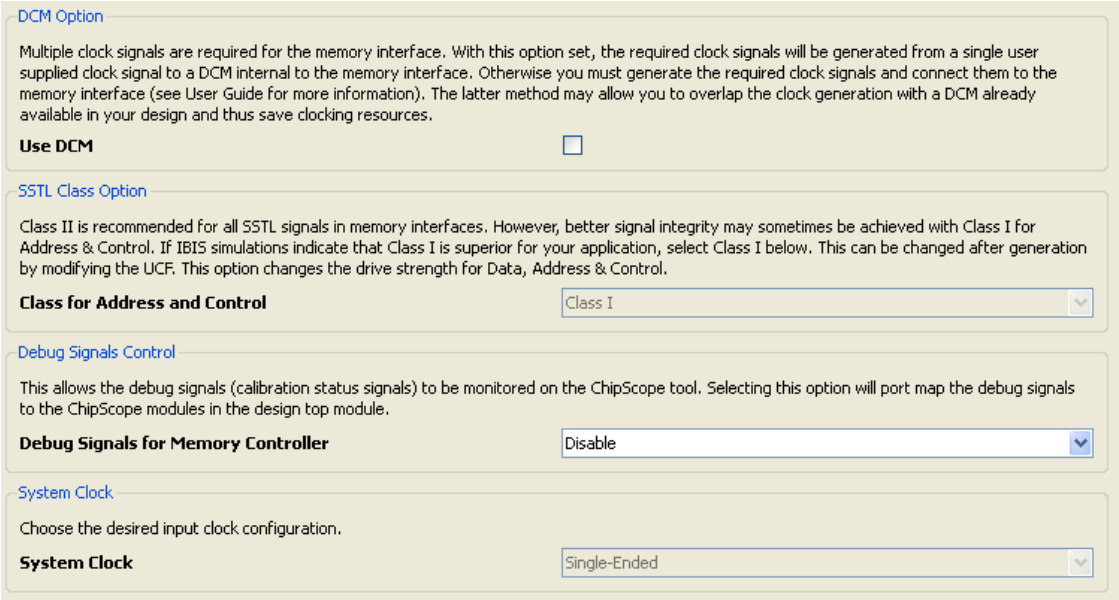

Ilustración 20: FPGA Options DDR.

6) **Pines Reservados (Reserved Pins):** No se hace uso de esta herramienta. En el UCF del diseño se especifican los pines de la DDR. Esta herramienta permite reservar pines, para que no sean utilizados por el MIG en el diseño del controlador.

7) **Selección del emplazamiento (Bank Selection):** Se selecciona el emplazamiento físico del control en la FPGA. A continuación se muestra el layout de la DDR y la FPGA:

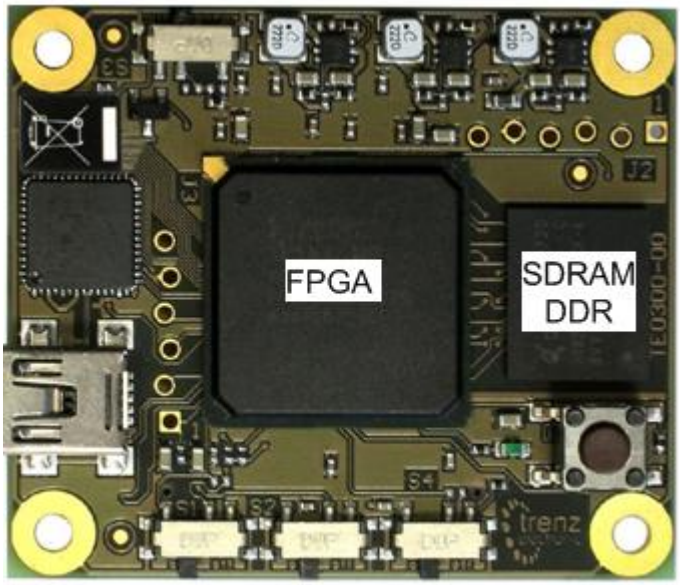

Ilustración 21: Layout físico DDR-FPGA

La DDR está conectada en el banco 1 de la FPGA. Por lo tanto generar el controlador en la lógica cercana al banco 1 facilitará el cumplimiento de los requisitos del fichero UCF.

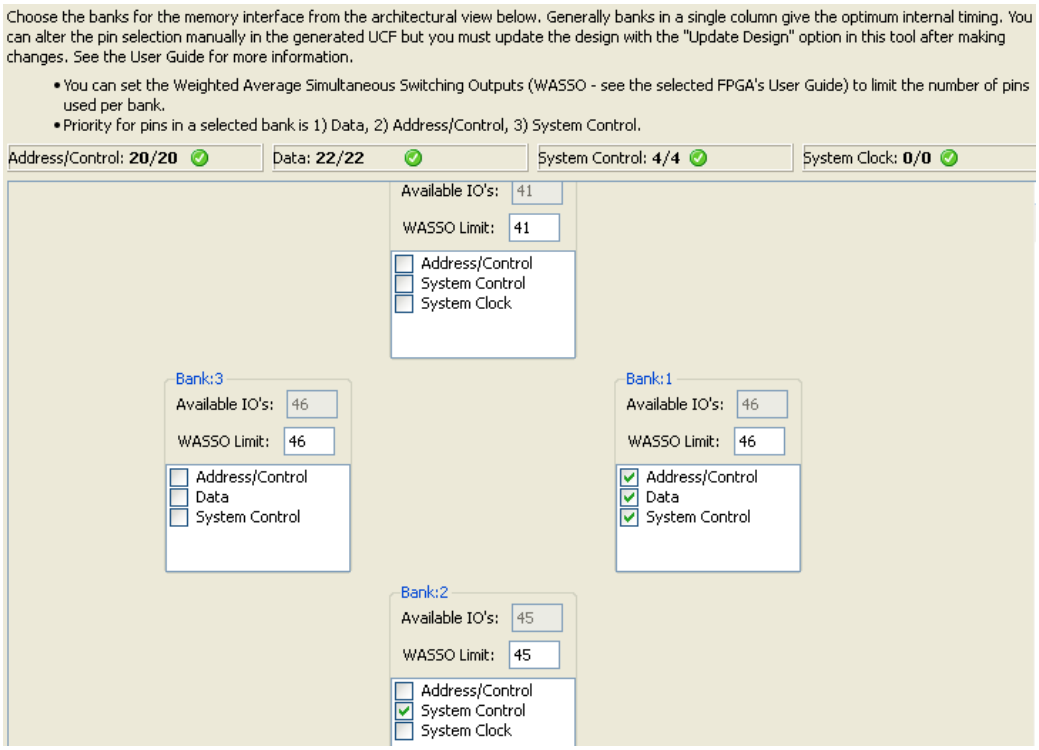

Ilustración 22: Selección del emplazamiento de la DDR

Los pasos finales son los mismos que los descritos en la creación del controlador de la DDR2 (sumario del diseño y licencia del modelo de simulación).

Finalmente se obtiene el controlador de la DDR SDRAM.

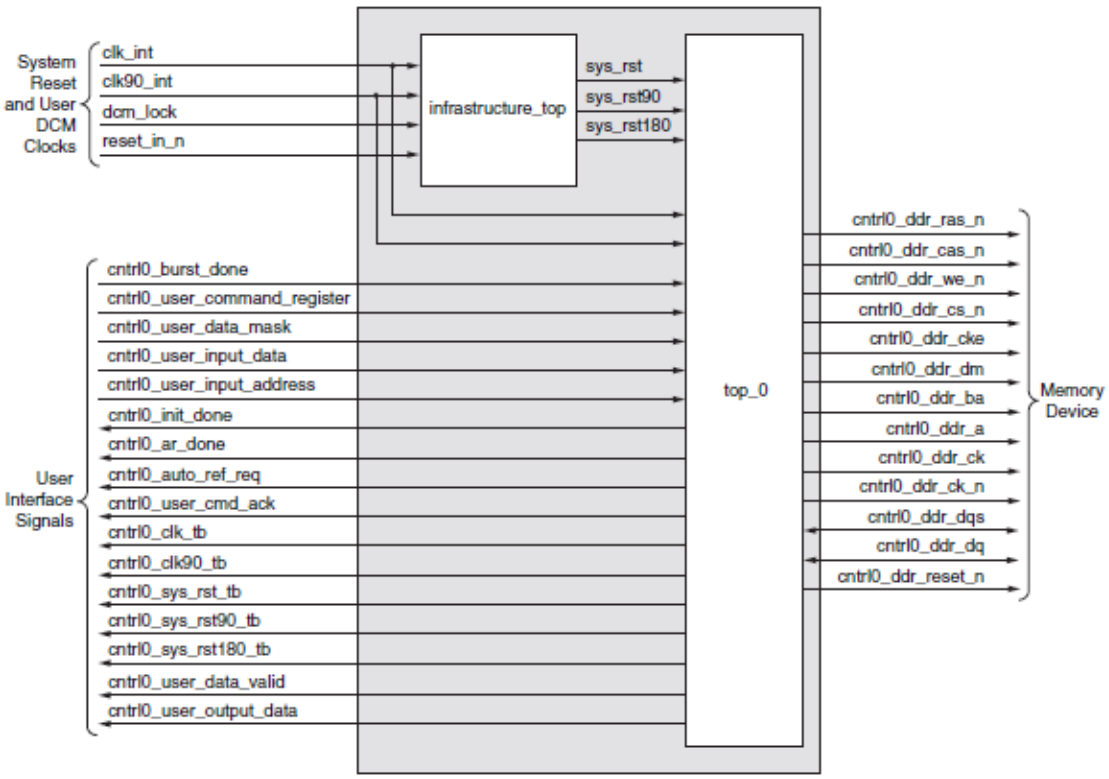

Ilustración 23: Diagrama de bloques del controlador de la DDR.

El controlador de la memoria solamente interactúa con el interfaz del controlador y con la memoria DDR, a excepción de la señal de reset_in, dcm_lock (conectadas a la señal de reset global del sistema) y los dos relojes de entrada del controlador de 80MHz, con uno de ellos desfasado un cuarto de ciclo respecto del otro.

Modificaciones en el UCF generado por el MIG:

Al igual que en el caso de la memoria DDR2 es necesaria la corrección y/o modificación de ciertos apartados del fichero UCF:

Modificación de rutas erróneas o inexistentes:

Modificación de errores debidos a especificaciones de rutas erróneas o inexistentes en Data Read, la etapa usada en la lectura de datos de la DDR2 (ver figura 24).

Incorrecto:

```
################################################################################
## Following are the MAXDELAY constraints on delayed rst_dqs_div net and fifo
## write enable signals. These constraints are required since these paths are
## not covered by timing analysis. The requirement is total delay on delayed
## rst_dqs_div and fifo_wr_en nets should not exceed the clock period.
################################################################################
NET "*/top_00/data_path0/data_read_controller0/rst_dqs_div"  MAXDELAY = 5000 ps;
NET "*/top_00/data_path0/data_read0/fifo*_wr_en*" MAXDELAY = 5000 ps;
```

Correcto:

```
################################################################################
## Following are the MAXDELAY constraints on delayed rst_dqs_div net and fifo
## write enable signals. These constraints are required since these paths are
## not covered by timing analysis. The requirement is total delay on delayed
## rst_dqs_div and fifo_wr_en nets should not exceed the clock period.
################################################################################
NET "*/top_00/data_path0/data_read_controller0/rst_dqs_div"
MAXDELAY = 5000 ps;
NET "*/top_00/data_path0/data_read_controller0/gen_wr_en[*]*fifo*_wr_en*"
MAXDELAY = 5000 ps;
```

Incorrecto:

```
################################################################################
## The MAXDELAY value on fifo write address should be less than clock period.
## This constraint is required since this path is not covered by timing
## analysis.
################################################################################
NET "*/top_00/data_path0/data_read0/fifo*_wr_addr[*]"
MAXDELAY = 10625 ps;
```

Correcto

```
################################################################################
## The MAXDELAY value on fifo write address should be less than clock period.
## This constraint is required since this path is not covered by timing
## analysis.
################################################################################
NET "*/top_00/data_path0/data_read_controller0/gen_wr_addr[*]*fifo*_wr_addr*"
MAXDELAY = 10625 ps;
```

Incorrecto:

```
NET "top_00/data_path0/data_read_controller0/gen_wr_en*fifo*_wr_en_inst/clk"
TNM_NET = "fifo_we_clk";
TIMESPEC "TS_WE_CLK" = FROM "dqs_clk" TO "fifo_we_clk"  5 ns DATAPATHONLY;
NET "top_00/data_path0/data_read_controller0/gen_wr_addr*fifo*_wr_addr_inst/
clk"
TNM_NET = "fifo_waddr_clk";
TIMESPEC "TS_WADDR_CLK" = FROM "dqs_clk" TO "fifo_waddr_clk"  5 ns
DATAPATHONLY;
```

Correcto:

```
NET "*/top_00/data_path0/data_read_controller0/gen_wr_en[*]*fifo*_wr_en*"
TNM_NET = "fifo_we_clk";
TIMESPEC "TS_WE_CLK" = FROM "dqs_clk" TO "fifo_we_clk"  5 ns DATAPATHONLY;
NET "*/top_00/data_path0/data_read_controller0/gen_wr_addr[*]*fifo*_wr_addr*"
TNM_NET = "fifo_waddr_clk";
TIMESPEC "TS_WADDR_CLK" = FROM "dqs_clk" TO "fifo_waddr_clk"  5 ns
DATAPATHONLY;
```

Ilustración 24: UCF, Modificaciones de ruta en etapa de lectura DDR

Modificaciones en el emplazamiento del controlador:

En este caso la placa sobre la que se trabaja no es específica de Xilinx. Por tanto, se debe modificar el emplazamiento de ciertos componentes del controlador en el UCF. Esto hará posible cumplir con los requisitos de tiempo, al cambiar los pines proporcionados por el MIG por los pines realmente conectados a la DDR.

"Dqs_delayed_col0" es la habilitación empleada para la captura de los 8 bits menos significativos del bus de datos de la memoria DDR.

Emplazamiento original (dqs_delayed_col0/five):

```
################################################################################################
## LUT location constraints for dqs_delayed_col0
################################################################################################
INST "*/top_00/data_path0/data_read_controller0/gen_delay[1].dqs_delay_col0/five" LOC = SLICE_X113Y73;
INST "*/top_00/data_path0/data_read_controller0/gen_delay[1].dqs_delay_col0/five" BEL = G;

INST "*/top_00/data_path0/data_read_controller0/gen_delay[0].dqs_delay_col0/five" LOC = SLICE_X113Y97;
INST "*/top_00/data_path0/data_read_controller0/gen_delay[0].dqs_delay_col0/five" BEL = G;
```

Emplazamiento modificado para (dqs_delayed_col0/five):

```
##############################################################################################
## LUT location constraints for dqs_delayed_col0
##############################################################################################
INST "*/top_00/data_path0/data_read_controller0/gen_delay[1].dqs_delay_col0/five" LOC = SLICE_X113Y72;
INST "*/top_00/data_path0/data_read_controller0/gen_delay[1].dqs_delay_col0/five" BEL = F;

INST "*/top_00/data_path0/data_read_controller0/gen_delay[0].dqs_delay_col0/five" LOC = SLICE_X113Y96;
INST "*/top_00/data_path0/data_read_controller0/gen_delay[0].dqs_delay_col0/five" BEL = F;
```

Se han tenido que cambiar los "slices" para la señal "rst_dqs_div". Esta señal sirve como habilitación (enable) para las FIFOs de lectura y para las FIFOS de los punteros de escritura después de pasar por una etapa de retardo. Un retardo incorrecto en esta señal puede significar la pérdida de los primeros datos en una lectura.

Emplazamiento original (rst_dqs_div):

```
##############################################################################
## Slice location constraints for delayed rst_dqs_div signal
##############################################################################
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/one" LOC = SLICE_X114Y85;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/one" BEL = F;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/two" LOC = SLICE_X114Y84;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/two" BEL = G;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/three" LOC = SLICE_X114Y85;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/three" BEL = G;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/four" LOC = SLICE_X115Y84;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/four" BEL = F;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/five" LOC = SLICE_X115Y84;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/five" BEL = G;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/six" LOC = SLICE_X115Y85;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/six" BEL = G;
```

Emplazamiento modificado (rst_dqs_div):

```
##############################################################################################
## Slice location constraints for delayed rst_dqs_div signal
##############################################################################################
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/one" LOC = SLICE_X114Y117;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/one" BEL = F;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/two" LOC = SLICE_X114Y116;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/two" BEL = G;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/three" LOC = SLICE_X114Y117;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/three" BEL = G;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/four" LOC = SLICE_X115Y116;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/four" BEL = F;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/five" LOC = SLICE_X115Y116;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/five" BEL = G;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/six" LOC = SLICE_X115Y117;
INST "*/top_00/data_path0/data_read_controller0/rst_dqs_div_delayed/six" BEL = G;
```

Finalmente, se han hecho algunas modificaciones menores como: modificación de los pines conexionados a la DDR, los protocolos de éstos, la inclusión de resistencias (pullup), etc.

Si se desea ampliar información sobre cualquier tema tratado en este apartado relativo al controlador de la DDR, se recomienda consultar el documento UG086 de Xilinx (pags 281-312).

El documento "Spartan-3E FPGA Industrial Micromodule User Guide" de Trenz contiene toda la información relativa a la placa sobre la que se ha realizado el presente diseño.

El funcionamiento de la memoria DDR se puede consultar en la hoja de características "512 Mb DDR SDRAM". Las características y el fabricante de la memoria se adjuntan al inicio del presente apartado.

A continuación se va a ver la arquitectura de las memorias DDR y DDR2 SDRAM y las principales características del controlador para la memoria.

2.1.4 Arquitectura de la DDR y DDR2 y principales características del controlador de la memoria

En este apartado se va a describir la arquitectura de las memorias DDR y DDR2. Además se va ha hacer una breve referencia al modo de funcionamiento del controlador de la memoria en los accesos a memoria.

Las memorias DDR y DDR2 empleadas en el proyecto son de 512 Mbits. La arquitectura de ambas memorias es la siguiente:

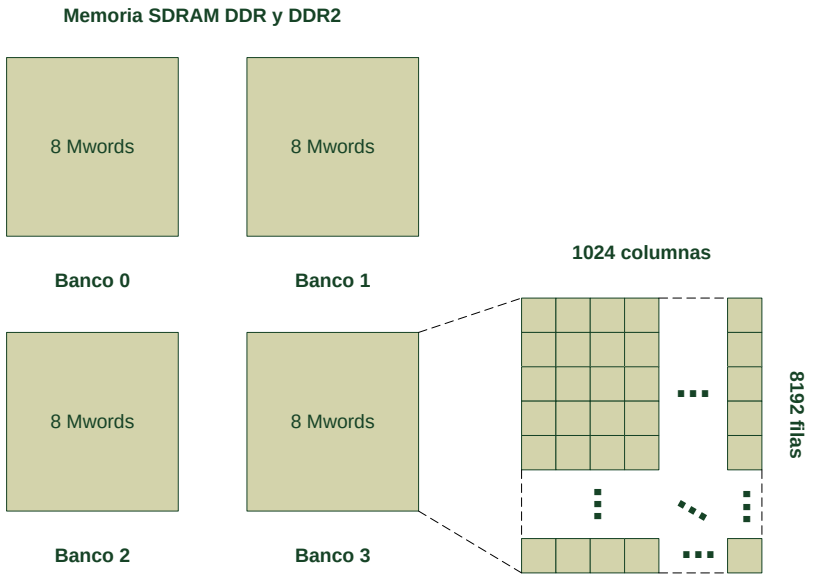

Ilustración 25: Arquitectura de la DDR y DDR2

Cada posición de la matriz en los bancos es de 16 bits (1 word). A continuación se detallan las principales características del controlador de la memoria en los accesos a memoria:

En un comando de escritura/lectura el controlador de la memoria únicamente puede acceder a un solo banco y a una sola fila (la primera dirección en un comando marca el banco y la fila a la que se accede), esto es, a un total de 1024 posiciones. En caso de ser necesario un cambio de fila y/o banco, se debe finalizar el comando actual de escritura/lectura e inicializar uno nuevo.

El controlador de la memoria va a solicitar un comando de refresco cada 7.5 microsegundos. En caso de estar ejecutándose un comando de escritura/lectura en el momento de solicitud de un comando de refresco, se deberá finalizar éste en un máximo de 20 ciclos de reloj.

El bus de direcciones a la entrada del controlador de la memoria es de 25 bits. El formato del bus de direcciones es:

$$ROW[24\ downto\ 12] + COLUMN[11\ downto\ 2] + BANK[1\ downto\ 0]$$

Row (fila): cada fila consta de 1024 columnas. En un comando de lectura/escritura sólo se puede acceder a una única fila, si se desea cambiar de fila durante un comando, se deberá finalizar éste e inicializar un nuevo comando. La selección de la fila está implícita en la primera dirección que se envía al controlador de la memoria al inicio de cada comando.

Column (columna): presentan un ancho de dato de 16 bits, es el único parámetro del bus de direcciones que puede variarse en un mismo comando de lectura o de escritura. Cada escritura/ lectura en memoria consta de 4 datos de 16 bits. El controlador de la memoria escribe/lee en la dirección proporcionada y en las 3 posiciones consecutivas. Para la generación de esas direcciones el controlador utiliza los dos bits menos significativos de la columna (bits 3 y 2), por tanto, solo se controlan 8 de los 10 bits.

Bank (banco): las dos memorias constan de un total de 4 bancos constituidos por 8192 filas y 1024 columnas cada uno. El procedimiento para cambiar de banco es análogo al descrito para cambiar de fila.

2.2 Interfaz para el controlador de la memoria DDR y DRR2

El objetivo del interfaz para el controlador de la memoria es simplificar la complejidad del controlador de cara al control de la FPGA y a la gestión de datos.

De esta manera se pretende que el interfaz para el controlador de la memoria, ejecute todos los procesos que requieran de escritura o lectura en memoria, como son: escritura de los datos muestreados, lectura de los datos muestreados, inicialización de la memoria, test de la memoria, así como otro tipo de procesos, como son el control del refresco de la memoria o el arranque de la memoria. En este capítulo el interfaz para el controlador será denominado comúnmente como "interfaz".

2.2.1 Requisitos funcionales del interfaz

A continuación se exponen los principales requisitos del interfaz:

Requisito_1: El interfaz debe ser totalmente compatible con el controlador de la memoria DDR y DDR2, además, el diseño debe de poder funcionar tanto con un reloj de 80 MHz como con un reloj de 125 MHz.

Requisito_2: Debe escribir en memoria al menos 10 datos de 32 bits cada microsegundo.

Requisito_3: Cuando se le solicite una lectura, debe proporcionar los datos leídos en memoria en el orden inverso en el que han sido escritos (del más nuevo al más antiguo), siendo el tamaño del bus de lectura de 16 bits.

Requisito_4: Cuando se inicie la lectura de la pila, debe de ser capaz de proporcionar todos los datos que en ese momento, se encuentren a la espera de escritura en memoria.

Requisito_5: Se requiere que sea capaz de realizar un test de la memoria que consista en escribir la zona de memoria usada con '1', verificar la lectura, y escribir la zona de memoria usada con '0' y verificar la lectura. Este test se iniciará cuando se solicite mediante un comando externo.

Requisito_6: El diseño del interfaz de memoria debe estar optimizado para requerir el número de recursos mínimo posible.

Requisito_7: Deberá cumplir con todas las especificaciones requeridas por el controlador generado por el programa MIG de Xilinx en el arranque, la lectura, la escritura y los tiempos de refresco de la memoria.

El diseño de un interfaz que cumpla con estos requisitos se ha realizado en dos etapas:

- En una primera etapa se diseño un interfaz denominado de acceso esporádico. Los accesos en memoria de escritura/lectura se realizan cada vez que el interfaz

recibe un número fijo de muestras, o bien, cada vez que el usuario lee un número fijo de datos del interfaz. Esto es, el interfaz siempre escribe/lee el mismo número de datos cada vez que accede a memoria. Este primer diseño se descartó debido a una mala relación área/velocidad.

- En la segunda etapa se diseño un interfaz denominado de acceso continuo. En este diseño el interfaz escribe en memoria durante el mayor tiempo posible. Si llega una nueva muestra, se incrementará el bus de direccionamiento de tal modo que la siguiente muestra se escriba en la siguiente dirección. Si no llega una nueva muestra, se machacará la posición marcada por el bus de direccionamiento. El objetivo de éste funcionamiento es el aumento significativo de la velocidad de procesamiento de datos del interfaz.

En el apartado 2.2.4 se realiza una comparativa entre los dos diseños del interfaz. La selección del diseño definitivo se basará en la relación área/velocidad.

2.2.2 Descripción del primer diseño: acceso esporádico

Introducción:

Se llama acceso esporádico a este diseño porque sólo se realizan escrituras en memoria cuando se han registrado n datos de 32 bits a escribir, donde n es un número que varía entre 4 y 16. A continuación se explican las razones de este acotamiento:

1) El mínimo de 4 escrituras es debido al requisito 1, en el que se especifica que el interfaz debe de ser compatible para las memorias DDR y DDR2. El controlador de la DDR2 tiene un burstlenght mínimo de 4. Se recuerda que burstlengh es un parámetro de diseño en la creación del controlador de la memoria con el MIG. Este parámetro indica el número de datos que se procesan por cada acceso de escritura o de lectura.

2) El máximo de 16 es debido al requisito 6. Por cada dato que se almacena a la espera de escritura en memoria se emplean 32 biestables. Ésto da un total máximo de biestables igual a 16*32 = 512 biestables. La principal ventaja de almacenar un gran número de datos antes de realizar una operación de escritura es la velocidad. El comienzo y finalización de un comando de escritura requiere una cierta cantidad de ciclos de reloj. Una vez que el comando de escritura está inicializado, se puede procesar un dato de 32 bits por cada ciclo de reloj. De lo anterior se deduce que si se quiere ganar velocidad en el procesamiento de las muestras, es mejor hacer menos comandos de escritura y realizar más accesos en cada comando, que hacer más comandos de escritura y realizar menos accesos en cada comando. El razonamiento es análogo para los comandos de lectura.

Diagrama de estados completo del interfaz:

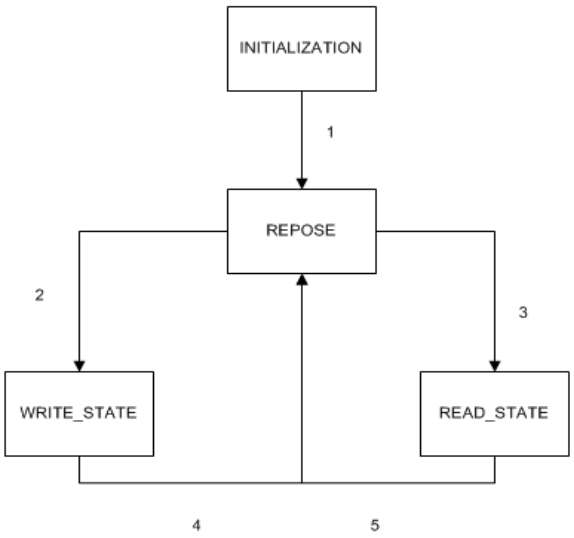

Ilustración 26: Diagrama de estados escritura esporádica

A continuación se expone con detalle cada estado del interfaz.

Initialization:

Una vez se desactiva la señal de reset global sobre el controlador, éste requiere de 200 microsegundos antes de poner a nivel alto los resets y de generar los relojes que actúan sobre el interfaz, este tiempo es requerido por la memoria para que se estabilicen los voltajes de referencia y la alimentación.

Cuando ha finalizado el arranque de la memoria, el interfaz envía el comando de inicialización al controlador ("010"). Una vez el controlador finaliza la inicialización se pone a nivel alto la señal init_done. A partir de éste momento la memoria está lista para inicializar cualquier comando de lectura y/o de escritura.

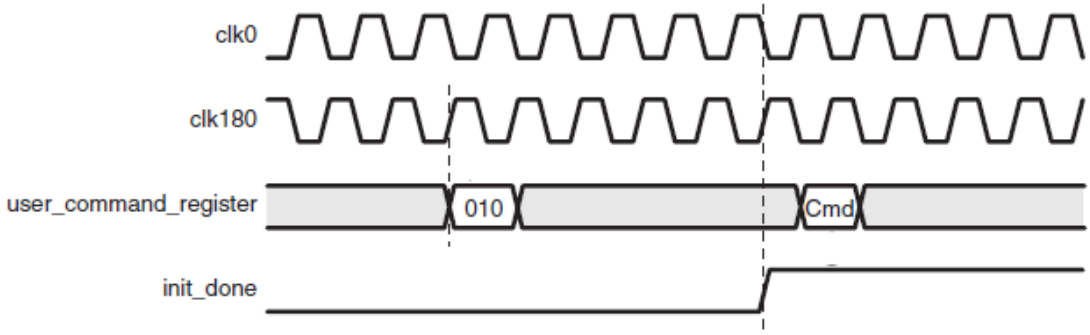

Ilustración 27: Inicialización de la memoria

Repose:

Durante el estado de reposo se comprueban las peticiones de escritura y de lectura solicitadas por la gestión de datos y el control de la FPGA respectivamente.

Se indica a la gestión de datos la disposición del interfaz para procesar un nuevo dato mediante una señal activa por nivel alto. Cada vez que la gestión de datos solicita una petición de escritura, mediante una señal de enable, se registra el dato y se comprueba el número de datos a escribir en memoria registrados. Cuando este número es igual a n se pasa a write_state, iniciándose un comando de escritura.

Cuando se solicita una petición de lectura al interfaz, mediante una señal de enable, se comprueba el número de datos útiles almacenados en los registros de lectura. Si es el último dato útil se pasa a read_state, para cargar los registros de lectura con nuevos datos. Si no es el caso, se incrementa el contador de control de datos leídos. Cada vez que se proporciona un nuevo dato tras una petición de lectura, se indica mediante una señal de enable.

El bus de datos en el que se proporcionan los datos leídos al control de la FPGA tiene un ancho de 16 bits, debido al requisito 3, por lo que en una sola lectura no se puede proporcionar un dato de 32 bits, siendo el orden en que se proporcionan los datos el siguiente (se recuerda que son datos de 32 bits): en primer lugar se proporcionarán los 16 bits de clasificación del dato, en segundo lugar los 16 bits de dato.

Durante el reposo, se puede solicitar el cambio de escritura a una lectura completa de la pila. En éste caso se prepara el bus de direcciones para cumplir con el requisito 3, es decir, que la memoria funcione como una LIFO.

Se puede dar el caso de una solicitud de paso de escritura a lectura y una existencia de datos útiles, registrados en los registros de escritura a la espera de ser escritos en memoria. De ser así, el sistema traspasa los datos de los registros de escritura a los registros de lectura, siendo imprescindible que el número de registros de escritura n sea igual o menor al número de los registros de lectura m. En el caso de no existir datos útiles en los registros de escritura se pasa al estado read_state.

Finalmente en un proceso aparte de la máquina de estados, se calcula el número de datos útiles en la pila de la memoria. Cuando se lee la pila completa se avisa al control de la FPGA, mediante una señal activa por nivel de la finalización de la lectura de todos los datos útiles de la pila.

A continuación se expone el diagrama de flujo en el estado de reposo:

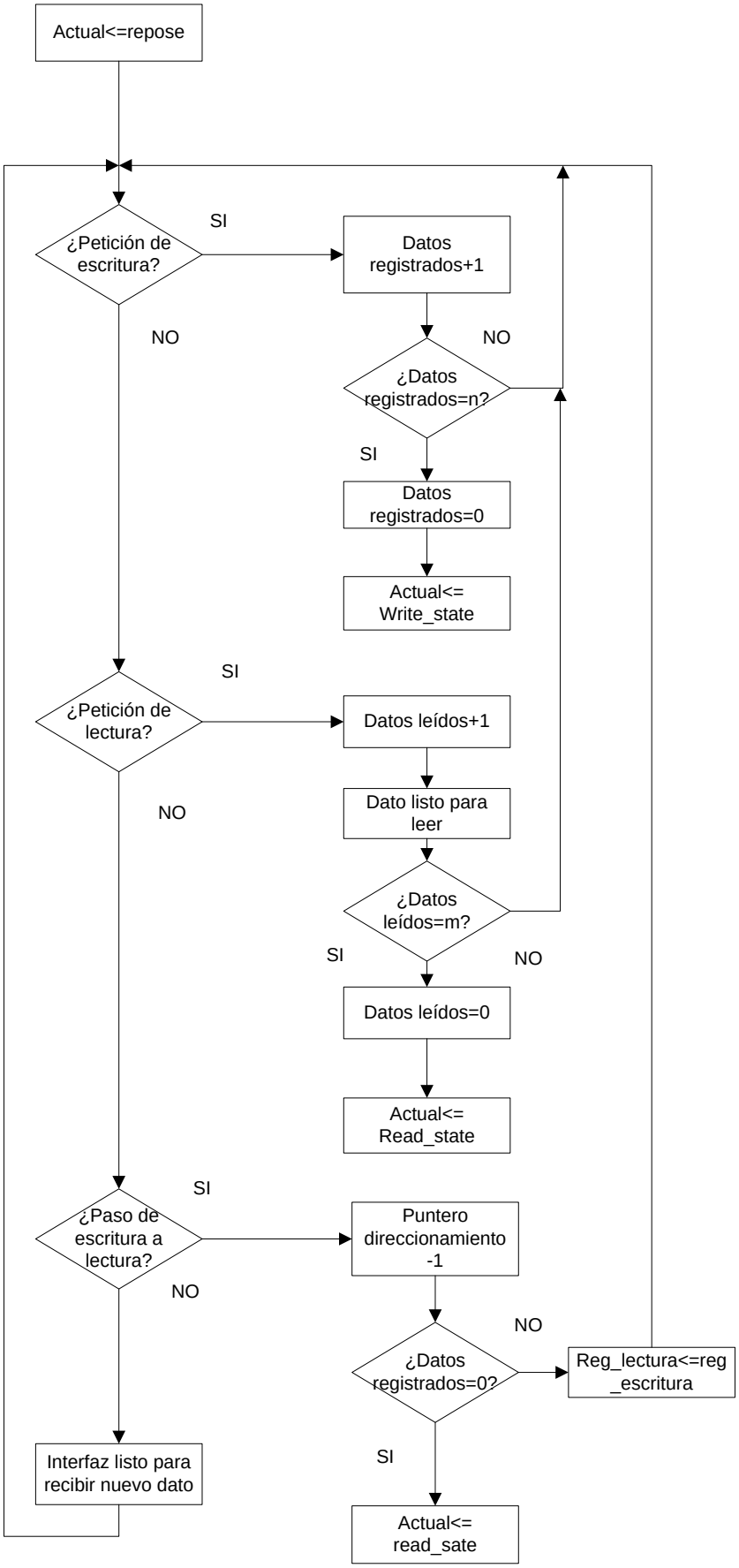

Ilustración 28: Diagrama de flujo reposo escritura esporádica

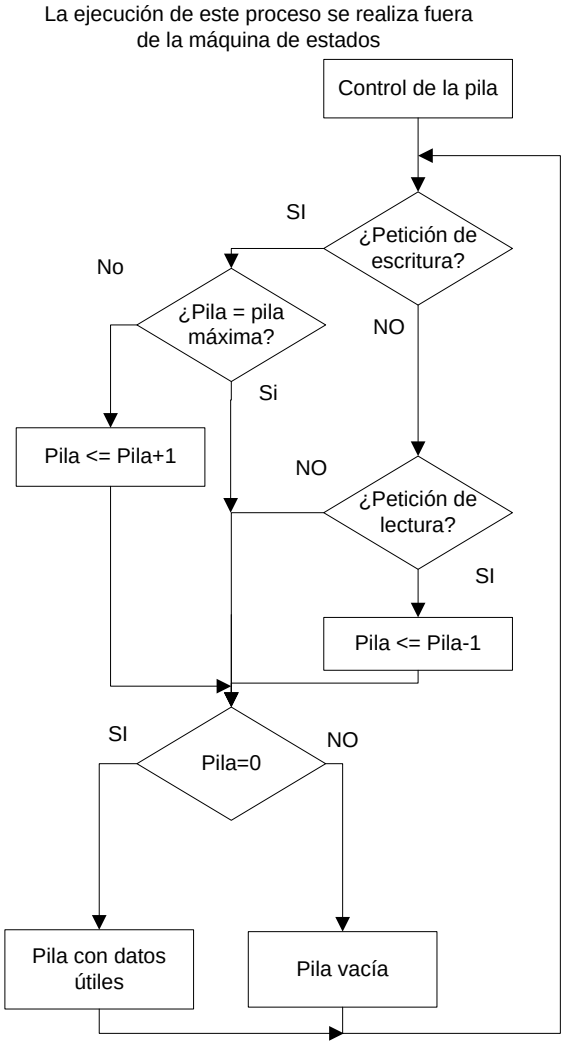

Ilustración 29: Control de la pila acceso esporádico

El diagrama de flujo anterior expone el cálculo que se realiza para indicar al control de la FPGA si hay datos útiles en la pila.

Write_state:

En este estado se inicia un comando de escritura, escribiéndose n datos de 32 bits.

El ancho del dato de la memoria es de 16 bits y el bus de datos de escritura entre el interfaz y el controlador es de 32 bits. Esto implica que en un solo ciclo de reloj se escriben 2 datos en memoria. El controlador es capaz de transformar el dato de 32 bits en dos datos de 16 bits.

El cronograma de un acceso a escritura y con n igual a 2 es el siguiente:

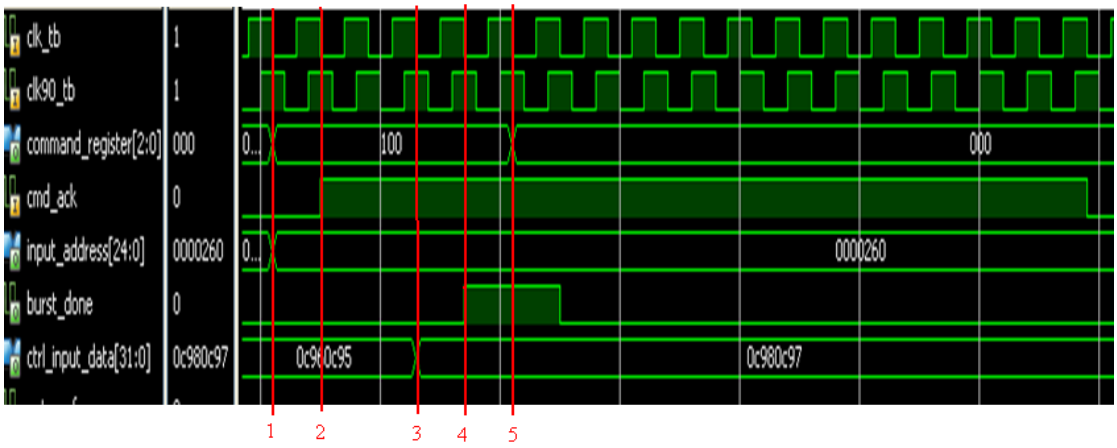

Ilustración 30: Comando de escritura

Clk_tb y clk90_tb son los dos relojes de del sistema desfasados 90º, command_register es el comando de control desde el interfaz al controlador de la memoria, cmd_ack indica al interfaz que el controlador de la memoria se halla en un comando de lectura/escritura, input_address es el bus de direcciones proporcionado por el interfaz, burst_done indica al controlador de la memoria la finalización de un comando de escritura/lectura y ctrl_input_data son los datos que el interfaz escribe en el controlador de la memoria.

1. Se envía el comando de escritura en el flanco de bajada de clk_tb. El primer dato (0C960C95) y dirección se ponen junto con el comando de escritura.

2. El controlador pone a nivel alto cdm_ack en el flanco de bajada de clk0, indicando la correcta recepción del comando de escritura. En el caso de que se esté ejecutando un refresco de la memoria, cmd_ack no se pondrá a nivel alto hasta la finalización del mismo. En cualquier otro caso en un ciclo de reloj estará a nivel alto.

3. El segundo dato (0C98OC97) y dirección se emplazan 3 ciclos de reloj después de que el controlador ponga a nivel alto cmd_ack, los subsiguientes datos (en el ejemplo no hay más datos) y direcciones se emplazan cada dos ciclos de reloj. En el ejemplo sólo se realiza un acceso de escritura.

4. Para terminar el comando de escritura se debe poner burst_done a nivel alto durante dos ciclos de reloj, tras completar la escritura de la última dirección.

5. Un ciclo de reloj después de poner a nivel alto burst_done, command_register toma el valor de “000”. Finalmente el controlador pone a nivel bajo la señal cmd_ack, indicando que se ha finalizado el comando de escritura.

Read_state:

Durante éste estado se realiza un comando de lectura, leyéndose m datos de 32 bits en memoria.

El cronograma de un comando de lectura con un solo acceso es el siguiente:

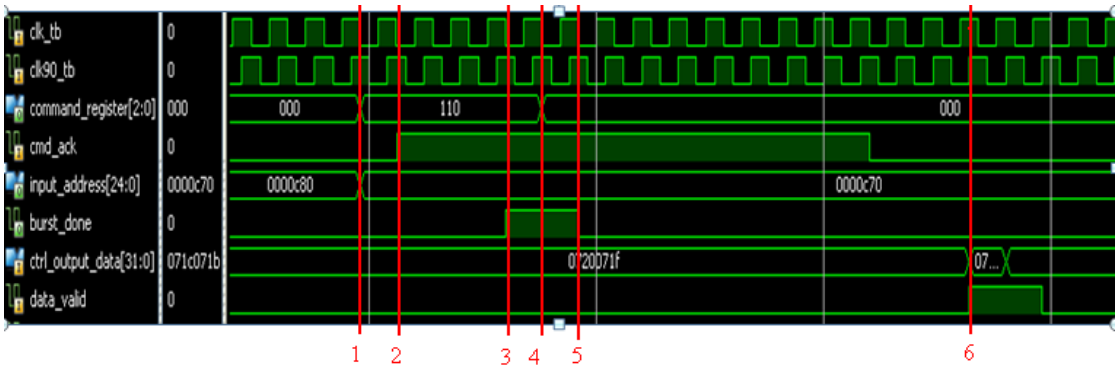

Ilustración 31: Acceso de lectura

Ctrl_output_data es el bus de datos de lectura proporcionado por el controlador de la memoria al interfaz y data_valid habilita la lectura de ctrl_output_data (una lectura válida por cada flanco de subida de clk90_tb en que data_valid esté a nivel alto). El resto de las señales del cronograma son análogas a las del cronograma de write_state.

1. Se emplaza el comando de lectura"110" junto con la primera dirección a leer en el flanco de bajada de clk_tb. La primera dirección se mantiene durante tres ciclos de reloj, las siguientes direcciones se emplazarán cada dos ciclos de reloj, en la diagrama solo se realiza un acceso de lectura (solo se lee en una dirección).

2. El controlador indica el inicio del comando de lectura poniendo a nivel alto cmd_ack en el flanco de bajada de clk_tb. En caso de que se esté ejecutando un comando de refresco, cmd_ack no estará a nivel alto hasta la finalización del mismo. En cualquier otro caso el controlador empleará un solo ciclo de reloj.

3. Para terminar el comando de lectura se pone burst_done a nivel alto durante dos ciclos de reloj tras tres ciclos de reloj de emplazar la primera dirección o dos ciclos de reloj de emplazar cualquier otra dirección.

4. Un ciclo de reloj después de poner burst_done a nivel alto se pone command_register a "000".

5. Cuando el controlador pone a nivel bajo cmd_ack se finaliza el comando de lectura.

6. La señal data_valid indica cuando es válida la lectura en el bus de datos. El bus de lectura y data_valid son síncronos con el flanco de subida de clk90_tb. En el diagrama se leen dos datos útiles de 32 bits (071C071B, el otro dato es el siguiente que no se puede apreciar en el cronograma). Se puede observar que es posible finalizar un comando sin que el controlador proporcione los datos leídos. Esto es debido al retardo interno del controlador. En el diagrama se observa un total de 17 ciclos de reloj. En cualquier caso no representa ningún inconveniente para poder iniciar nuevos comandos de lectura y/o de escritura.

Una vez se han descrito los estados del interfaz, se muestran las transiciones del diagrama de estados.

Transiciones

1. Una vez finalizado el reset, el controlador espera 200 microsegundos hasta que arranca la memoria, una vez la memoria arranca lo indica al interfaz poniendo a nivel alto la señal init_done, pasando el interfaz al estado de reposo.

2. Cuando llega un nuevo dato para escribir en la memoria, el interfaz comprueba el número de datos que tiene registrados. Si son n datos (incluyendo el dato que acaba de llegar), se pasa al estado write_estate donde se inicia un comando de escritura, escribiéndose los datos almacenados en los registros. Se recuerda que n varía de 4 a 16 y se puede variar en función de la velocidad que se quiera conseguir.

3. Cuando se solicita al interfaz la lectura de un dato, éste comprueba el número de datos que tienen disponibles en sus registros de lectura. Una vez que proporciona el último dato que tiene registrado, se inicia un comando de lectura (read_state), donde se leen n datos (de nuevo el rango de n es 4 a 16).

4. Cuando finaliza el comando de escritura de n datos en memoria, el interfaz vuelve al estado de reposo.

5. Cuando finaliza el comando de lectura de n datos en memoria, el interfaz vuelve al estado de reposo.

A continuación se detallan los refrescos de la memoria, las limitaciones del diseño y los puertos y bloque del inetrfaz.

Refrescos de la memoria:

Las memorias DDR y DDR2 al ser memorias volátiles, necesitan un refresco de los datos cada 7.5 microsegundos aproximadamente. Cuando se está ejecutando un comando de refresco no se pueden iniciar comandos de lectura y/o de escritura.

El controlador emplea una señal activa por nivel para solicitar un comando de refresco. Si no se están ejecutando comandos de lectura y/o de escritura el intefaz procede al refresco inmediato de la memoria. En caso contrario se espera hasta la finalización de dichos comandos. El controlador indica la finalización del refresco mediante una señal de habilitación.

Limitaciones del diseño:

1. En el paso de escritura a lectura, antes de realizar peticiones de lectura, se debe dejar un tiempo muerto de al menos 100 ciclos de reloj para que se finalicen los posibles comandos de escritura o refrescos y se prepare el sistema para la lectura.

2. Se debe mantener un tiempo mínimo de 50 ciclos de reloj entre peticiones de lectura para la finalización de los posibles comandos de lectura. Aunque el

controlador de la memoria si es capaz de iniciar nuevos comandos de lectura, la lógica del interfaz no está preparada para ello.

3. Se debe mantener un tiempo mínimo de 50 ciclos de reloj para el paso de escritura a lectura, para finalizar los posibles comandos de lectura en memoria.

4. No se consigue el requisito de velocidad necesaria a un coste de área razonable (apartado 2.2.4).

Puertos y bloque del interfaz:

Puerto	Tipo	Descripción
burst_done	Out	Esta señal se usa para terminar comandos de lectura y de escritura, poniéndose a nivel alto durante 2 ciclos de reloj (solo para un burst lenght=4). Desde el interfaz al controlador.
init_done	In	Señal activa por nivel, indica inicialización de la memoria completa poniéndose a '1'. Desde el controlador al interfaz.
ar_done	In	Enable, indica finalización del comando de refresco desde el controlador al interfaz.
data_valid	In	Cuando está a nivel alto, indica que ctrl_output_data es válido. Desde el controlador al interfaz.
auto_ref_req	In	Petición de refresco del controlador al interfaz, aproximadamente cada 7.5 microsegundos.
cmd_ack	In	Indica acceso en memoria cuando está a nivel alto.
command_register [2 downto 0]	Out	Comandos del interfaz para el controlador de la memoria: "000" => nada (usado en los refrescos) "010" => inicialización de la memoria "100" => escritura en memoria "110" => lectura en memoria Otros => reservado
clk_tb	In	Reloj, del controlador al interfaz.
clk90_tb	In	Reloj desfasado 90 grados respecto de clk_tb.
reset_tb	In	Reset del controlador al interfaz, síncrono con el flanco de subida de clk_tb, activo por nivel alto.
reset90_tb	In	Reset del controlador al interfaz, síncrono con el flanco de subida de clk90_tb, activo por nivel alto.
reset180_tb	In	Reset del controlador al interfaz, síncrono con el flanco de bajada de clk_tb, activo por nivel alto.
ctrl_output_data [31 downto 0]	In	Bus de lectura entre el interfaz y el controlador.
ctrl_input_data [31 downto 0]	Out	Bus de escritura entre el interfaz y el controlador.
input_address [24 downto 0]	Out	Bus de direccionamiento de la memoria: [24 downto 12] => Row [11 downto 2] => Column [1 downto 0] => Bank
enable_write	In	Enable de petición de escritura de la gestión al interfaz.

Puerto	Tipo	Descripción
input_data [31 downto 0]	In	Bus de escritura entre el interfaz y la gestión.
load_write	Out	Interfaz listo para recibir escrituras de la gestión.
enable_read	In	Enable de petición de lectura desde el control de la FPGA al interfaz.
load_read	Out	Enable de dato listo para su lectura desde el interfaz al control de la FPGA.
output_data [31 downto 0]	Out	Bus de lectura entre el interfaz y el control de la FPGA.
enable_start	In	Enable de paso de lectura a escritura desde el control de la FPGA al interfaz..
dato_valido	Out	Indica si se ha leído todos los datos escritos en memoria.
ref_ddr2	Out	Señal para medir el tiempo de duración de un refresco.

Ilustración 32: Puertos interfaz acceso esporádico.

Bloque interfaz:

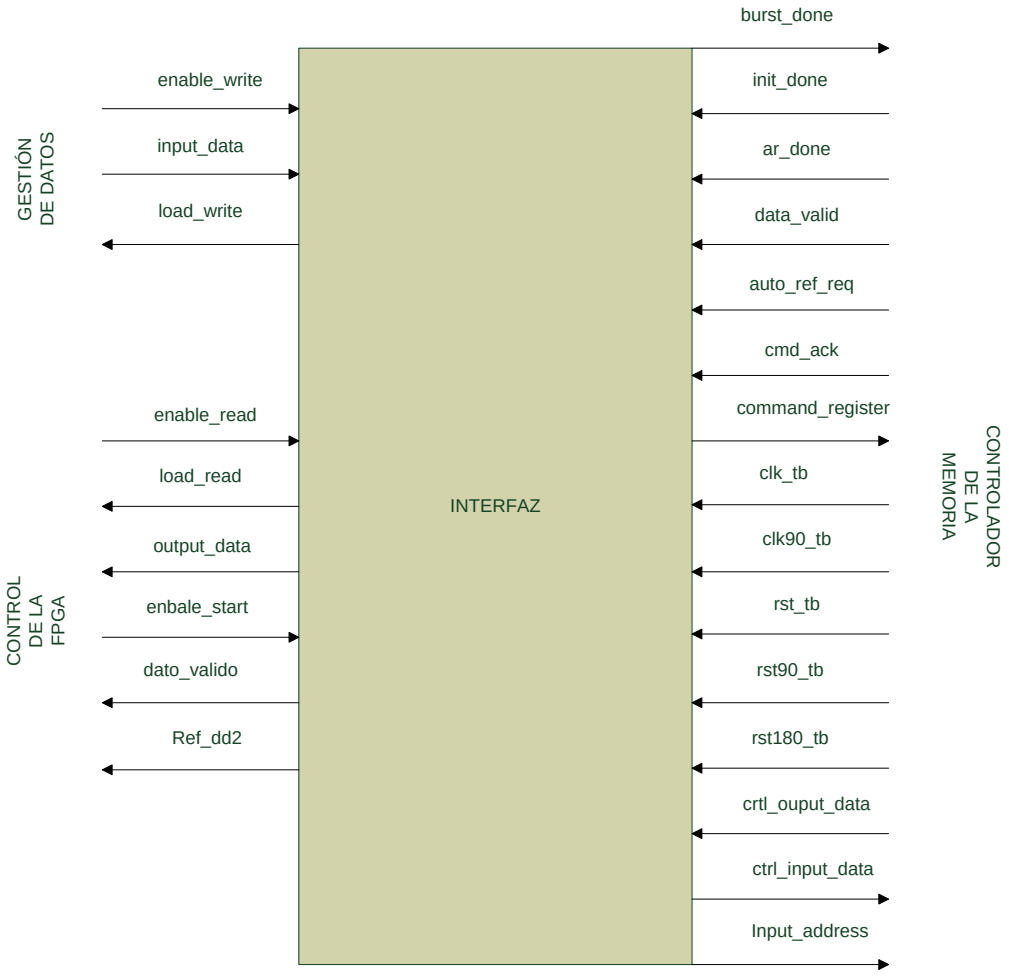

Ilustración 33: Bloque interfaz acceso esporádico

2.2.3 Descripción del segundo diseño: acceso continuo

Se denomina acceso continuo a este diseño debido a que durante el estado de escritura de muestras en memoria, el interfaz realiza comandos de escritura con el mayor número de accesos posibles.

Es importante aclarar que el acceso continuo en memoria solo se realiza en los comandos de escritura, no así en los comandos de lectura. Esta diferencia es debida a que el principal motivo de realizar un acceso continuo es el aumento de la velocidad de escritura. La lectura en memoria no presenta ningún problema de tiempo, por tanto, se cumple con las especificaciones de tiempo y de lógica impuestas por los requisitos de diseño.

Las principales ventajas de un acceso continuo en memoria son las siguientes:

1. Se minimizan las pérdidas de tiempo en la inicialización y finalización de accesos a memoria, cumpliéndose con el requisito 2.

2. Se ahorran los biestables encargados de registrar los datos, por tanto, cada dato se procesa antes de que llegue el siguiente, cumpliéndose con el requisito 6.

A continuación se expone el funcionamiento del módulo interfaz con acceso continuo en memoria:

Diagrama de estados completo del interfaz:

Se ha hecho un diseño con dos máquinas de estados. La de más alto nivel es la máquina de estados de test, con la que se inicializa la pila y se verifica la memoria.

Hay dos tipos de test a los a los que se puede someter al interfaz:

1. Test completo de la memoria (test_1): se inicializa la pila y se verifica la memoria. Consiste en la escritura de “0xFFFF” y de “0x0000” para en todas las posiciones de la memoria y la correspondiente verificación del dato escrito en memoria mediante una lectura completa.

2. Inicialización de la pila (test_0): únicamente se escribe “0x0000” en toda la pila. La utilidad de este test respecto del anterior, es un ahorro de tiempo considerable, si no es necesario verificar la escritura en memoria o si en un test anterior ya se ha verificado la memoria.

El diagrama de estados es el siguiente.

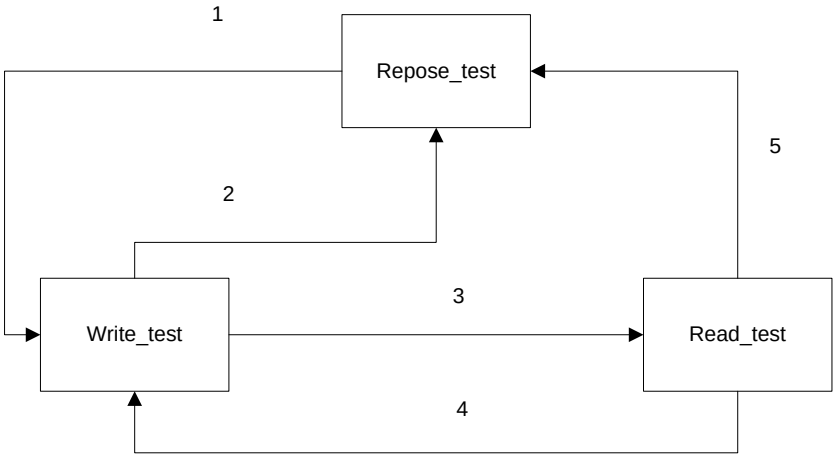

Ilustración 34: Diagrama de estados 1, escritura continua

La descripción de los estados es la siguiente de la Ilustración 34:

Repose_test:

El interfaz funciona con normalidad, es decir, es capaz de procesar datos para ser escritos en memoria o ser leídos de la memoria.

Write_test:

Se realizan los comandos de escritura necesarios durante el test_0 y el test_1. Se enmascaran los puertos de control de la escritura y se genera la lógica de control necesaria para escribir en memoria 0x"0000" o 0x"FFFF".

Read_test:

Se realizan comandos de lectura durante la realización del test_1. En los comandos de lectura, para optimizar el tiempo de realización del test_1, se realizan accesos continuos en memoria. Se enmascaran los puertos de control de la lectura y se genera la lógica de control necesaria para leer en memoria y verificar la correcta lectura de los datos.

A continuación se expone el diagrama de flujo de funcionamiento de los tests:

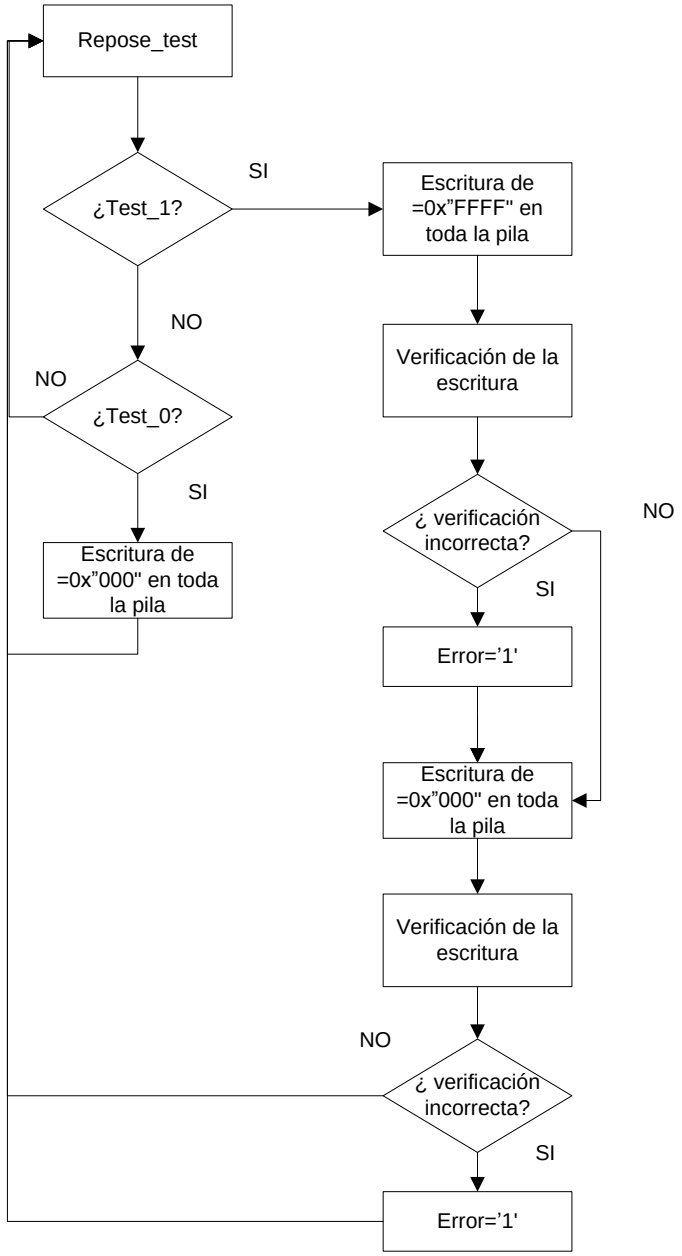

Ilustración 35: Diagrama de flujo de test

La asignación del error se hace durante la verificación, no después de la verificación. Por lo tanto, se evalúa la asignación de error cada vez que se lee un nuevo dato.

Una vez se conoce el funcionamiento de los estados de test, se describen las transiciones entre los estados.

Transiciones en el diagrama de estados de test:

1. El interfaz se halla en un estado en el que no se ejecuta ningún test. Se inicia un test tras una petición por parte del control de la FPGA, tanto del test_0 como del test_1.

2. El interfaz se halla en un estado de escritura en memoria. Tras la finalización de la escritura de "0x0000" en la pila se finaliza el test_0.

3. El interfaz se halla en un estado de escritura en memoria. Tras finalizar la escritura en memoria, se procede a la verificación del dato escrito durante la realización del test_1

4. El interfaz se halla en un estado de verificación de la pila. Se inicia la escritura en memoria del dato "0x0000" tras verificar la escritura del dato x"FFFF" durante el test_1.

5. El interfaz se halla en un estado de verificación de la pila. Se pasa a un estado de reposo tras verificar la escritura de "0x0000" en todas las posiciones de memoria.

Es importante aclarar que durante la realización de los test, el interfaz no puede ejecutar ninguna otra funcionalidad.

Por otro lado, se tiene la máquina de estados que gobierna al controlador de la memoria, generándose los comandos de lectura, escritura e inicialización correspondientes. A continuación se expone el diagrama de estados del interfaz:

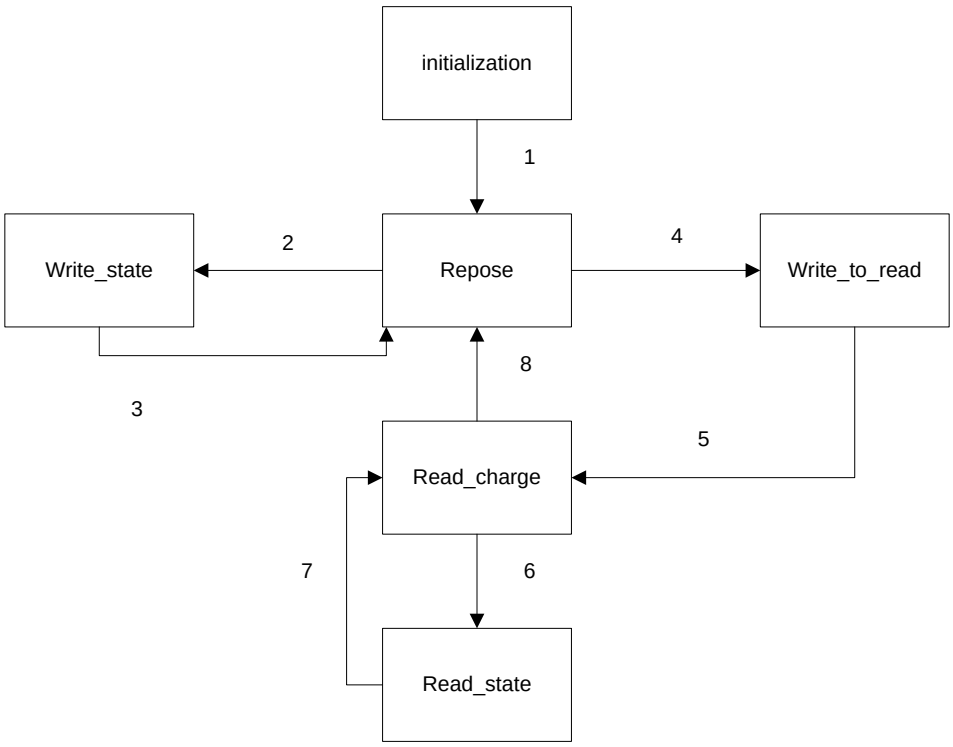

Ilustración 36: Diagrama de estados acceso continuo

La descripción de los estados es la siguiente:

Initialization:

El estado inicialización es completamente análogo al estado de inicialización del interfaz de acceso esporádico.

Repose:

El estado de repose es, básicamente, un estado de apoyo a write_state y a la correcta sincronización del bus de direcciones.

Cuando el sistema pasa de un estado de escritura a un estado de lectura, desde repose se preparará el bus de direcciones para que apunte sobre la dirección de la memoria con el último dato en ser escrito. A continuación se pasará al estado de write_to_read. A partir de este momento, no se retornará a repose hasta que el sistema pase a un estado de escritura o se solicite la iniciación de un test sobre el interfaz.

Cuando el sistema está funcionando en el estado de escritura, el interfaz va a estar continuamente ejecutando comandos de escritura en write_state con la mayor duración posible en el tiempo, es decir, solo se van a finalizar dichos comandos ante peticiones de refresco o cuando se complete la escritura de una fila de la memoria, haciendo necesaria la precarga de una nueva fila. Ante este funcionamiento del sistema, en el estado de repose se van a ejecutar los comandos de refresco y las precargas de las direcciones. Una vez que finalicen éstos procesos, se procederá inmediatamente al inicio de un nuevo comando de escritura en write_state.

Repose proporciona un apoyo a la correcta sincronización del bus de direcciones al recibir una petición de test, poniendo el bus de direcciones a 0.

A continuación se expone el diagrama de flujo de repose:

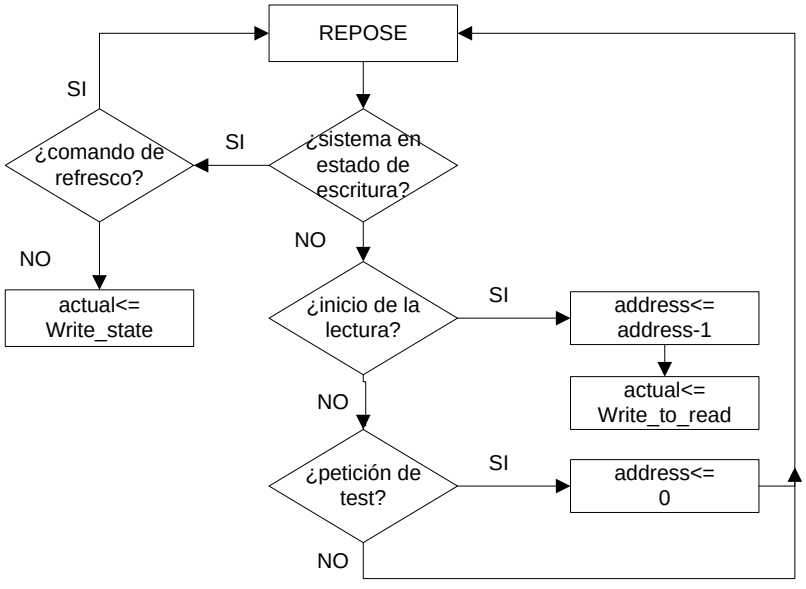

Ilustración 37: Diagrama de flujo repose, acceso continuo

Write_state:

En write_state se ejecutan los comandos de escritura sobre el controlador. La lógica de control realizada sobre los comandos de escritura, está descrita en el capítulo 3.2.2 (write_state).

El diseño de "acceso continuo" requiere que las muestras que se escriben en memoria sean procesadas en un solo ciclo de reloj, es decir, que se pueda escribir sobre el controlador un dato de 32 bits por cada ciclo de reloj. Debido a esto, es imprescindible que el comando de escritura se inicie (y sea estable) antes de que llegue una muestra. De ésta manera se podrá escribir un dato por cada ciclo de reloj. Si tuviéramos que iniciar y finalizar un comando de escritura por cada muestra que llega al interfaz, el coste en tiempo sería inasumible para el cumplimiento de las especificaciones de diseño.

A continuación se muestra en el siguiente cronograma, el comportamiento del interfaz ante la llegada de una nueva muestra con un comando de escritura estable:

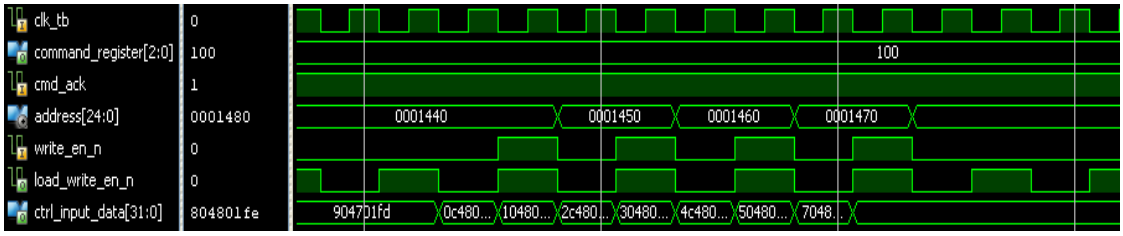

Ilustración 38: Escritura en memoria acceso continuo

El principal inconveniente de realizar comandos de escritura durante el mayor tiempo posible, es que se escriben datos en memoria en cada ciclo de reloj lleguen o no lleguen muestras. La solución a éste problema es implementar una lógica que actúe sobre el bus de direcciones, de tal manera que siempre que no llegue una muestra se estará sobrescribiendo la dirección consecutiva a la última posición escrita con una muestra. En el caso de que llegue una muestra, se escribe la muestra en la dirección que previamente estaba siendo sobrescrita y se incrementa el bus de direcciones.

Si se tiene en consideración que en cada acceso de escritura se gestionan 2 muestras de 32 bits se debe hacer una modificación a la lógica citada anteriormente. El bloque de gestión de datos sólo escribirá muestras en el interfaz siempre que esté en condiciones de enviar un número par de muestras y sea capaz de escribir éstas en cada ciclo de reloj de manera consecutiva.

Como se ve en el diagrama de señales, la gestión de datos escribe 8 muestras de 32 bits (ctrl_input_data) en el interfaz (en este caso, el bus de entrada al interfaz se conecta directamente a la entrada del controlador). Con la señal write_en_n se indica que se han escrito dos muestras en el interfaz, incrementándose el bus de direcciones (de éste modo se evita que se sobrescriban las muestras escritas en memoria). La señal load_write_en indica la disposición del interfaz de recibir 2 nuevas muestras (activa

por nivel alto). Finalmente, en los ciclos de reloj en los que no se reciben muestras, no se incrementa el bus de direcciones.

Existen 3 causas por las que se puede finalizar un comando de escritura:

1. El inicio de la lectura en memoria, ya sea porque el sistema inicia la lectura de la pila o porque durante un test se inicia la verificación de la escritura.

2. Se completa la escritura en una fila, por lo que se debe iniciar un nuevo comando de escritura cargando una nueva fila (en caso de no finalizarse el comando de escritura, se sobrescribe la fila actual).

3. La petición de un comando de refresco por parte del controlador.

En el siguiente diagrama de señales se muestra el funcionamiento de un comando de refresco en la memoria:

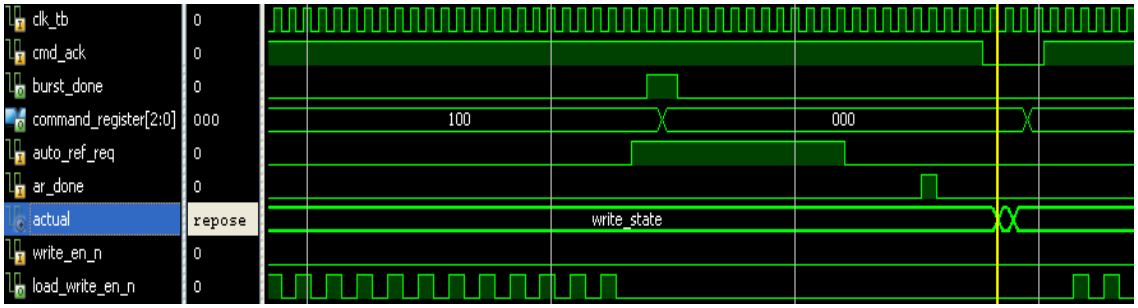

Ilustración 39: Comando de refresco

Se puede observar que cuando el controlador solicita una petición de refresco (auto_ref_req), el interfaz finaliza su disposición para recibir nuevas muestras (load_write_en) y finaliza el actual comando de escritura. Desde repose se verifica que el comando de refresco ha terminado (enable de ar_done) y la máquina de estados pasa a write_state iniciándose un nuevo comando de escritura.

Write_to_read:

Write_to_read es un estado cuya funcionalidad radica en el paso de escritura a lectura por parte del sistema.

Existe la posibilidad de que haya muestras a la espera de escribirse en el interfaz por parte de la gestión de datos y que a la vez el control de la FPGA solicite el inicio de la lectura de la pila. En el caso de que se de esta situación, el interfaz debe proporcionar al control de la FPGA todos los datos que esperen a ser escritos en la gestión de datos.

En este caso se implementa una lógica en la que es el interfaz el que lee las muestras almacenadas en la gestión de datos y los coloca en el bus de lectura del control de la FPGA.

Durante la ejecución de los test no interesan las muestras que puedan estar almacenadas en la gestión de datos por lo que se pasa a read_chargue de forma automática.

En el siguiente diagrama de flujo se expone la lógica implementada en write_to_read:

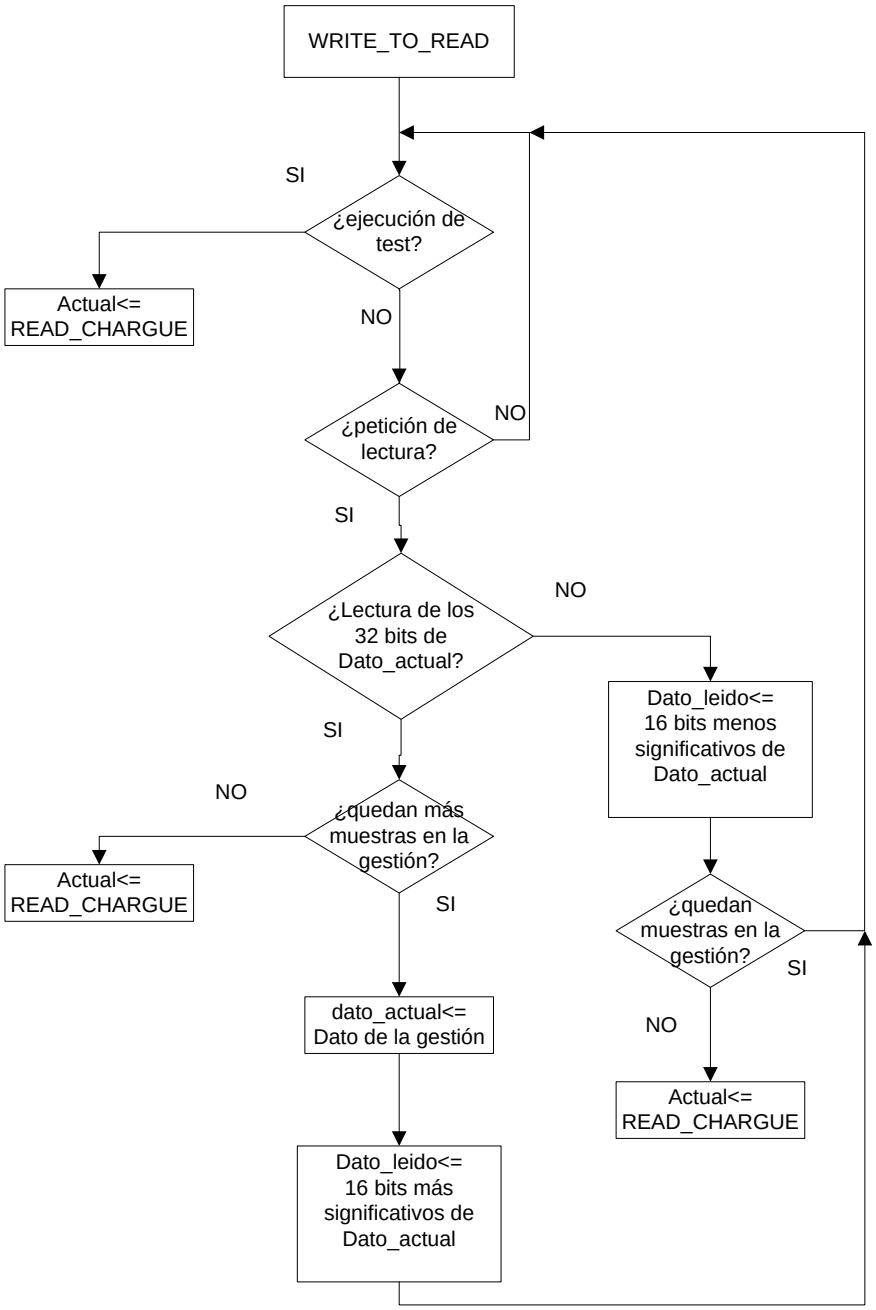

Ilustración 40: Diagrama de flujo write_to_read

El interfaz registra las muestras de 32 bits de la gestión de datos, pero el bus de lectura entre el interfaz y el control de la FPGA es de 16 bits. Esto implica que cada muestra leída en la gestión de datos proporciona dos lecturas para el control de la FPGA. Ante una petición de lectura primeramente se leen los 16 bits más significativos, y

finalmente cuando se recibe una nueva petición de lectura se leen los 16 bits menos significativos de la muestra.

Read_chargue:

Read_chargue es el estado en el que se gestionan las peticiones de lectura, proporcionándose al control de la FPGA los datos registrados durante los comandos de lectura ejecutados en read_state.

La lógica implementada en la gestión de las peticiones de lectura se muestra en el siguiente diagrama de flujo:

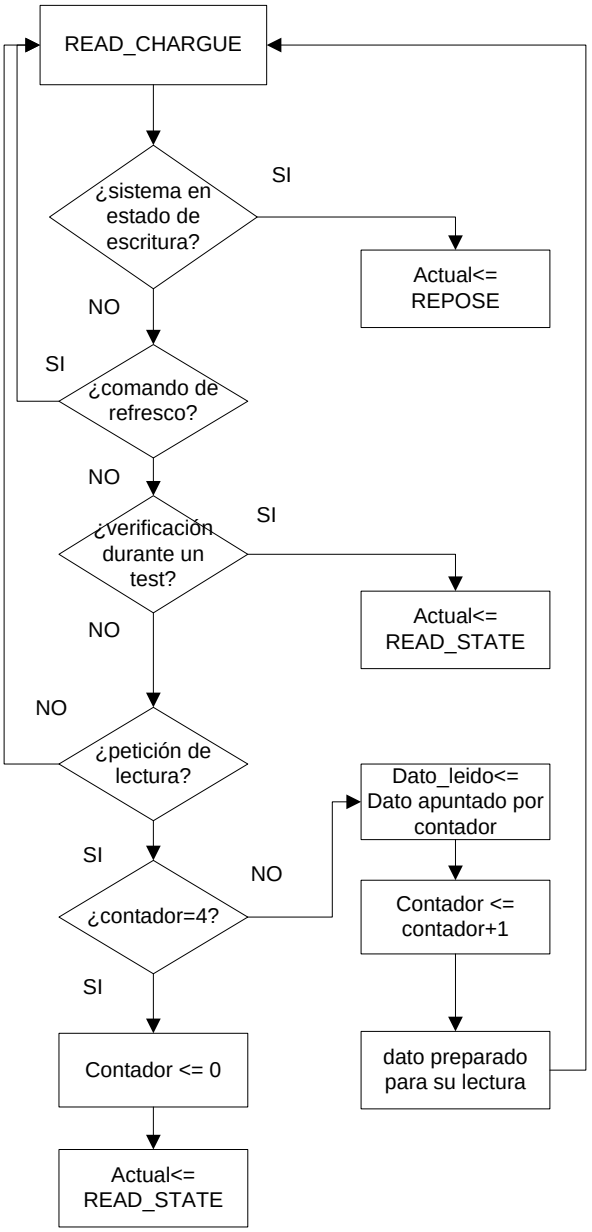

Ilustración 41: Diagrama de flujo de read_chargue

Si se está ejecutando la verificación de un test, se procede a un acceso continuo en memoria para finalizar la verificación en el menor tiempo posible (en este caso el funcionamiento de la lectura es similar al de write_state).

En el caso de que no se esté ejecutando un test, se dispone de 2 registros de 32 bits en los que están almacenados los datos que previamente se han leído de la memoria durante read_sate.

Para la lectura de los datos, se emplea un puntero que direcciona los datos de los registros de lectura. En el diagrama de flujo se le ha denominado contador (control_read en el código VHDL). Si se tiene en cuenta que el bus de lectura entre el control del la FPGA y el interfaz es de 16 bits y que el bus de lectura entre el interfaz y el controlador es de 32 bits, los datos proporcionados por el interfaz ante una petición de lectura, son los siguientes:

Nº de peticiones de lectura	**Valor de "contador"**	**Registro direccionado**	**Bits leídos del registro**
1	1	0	31 downto 16
2	2	0	15 downto 0
3	3	1	31 downto 16
4	4	1	15 downto 0

Tabla 1: Direccionamiento de los registros de lectura

Una vez se leen los 4 datos de 16 bits almacenados en los 2 registros y el usuario solicita la lectura de nuevos datos, se pasa a read_state, iniciándose un comando de lectura donde se precargan los registros. A continuación se muestra el diagrama de flujo de la precarga de los registros.

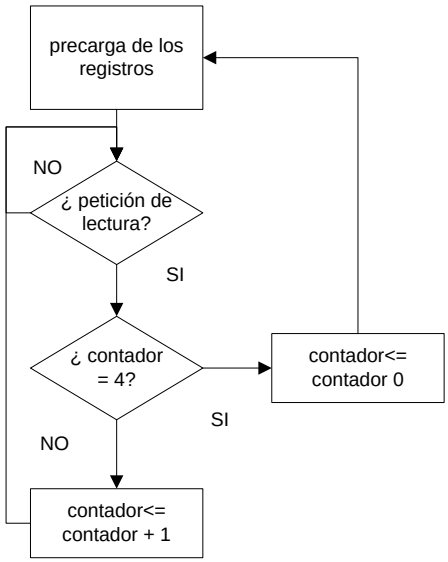

Ilustración 42: Precarga de los registros de lectura

Read_state:

Read_state es el estado donde se realizan los comandos de lectura sobre la memoria.

La lógica de control de los comandos de lectura es análoga a la descrita en el estado read_state del acceso esporádico.

Existen dos modos de funcionamiento para los comandos de lectura. Modo ejecución de un test y modo lectura de pila.

Modo Ejecución de un test: Si se está ejecutando una verificación de la memoria durante un test, en cada comando de lectura se verificará una fila completa hasta completar la verificación de toda la pila. La única excepción que se contempla, es la petición de un comando de refresco por parte del controlador durante la ejecución de un comando de lectura, en cuyo caso se finalizará el presente comando de lectura sin finalizar la lectura de la pila, se ejecutará un comando de refresco y posteriormente se completará la verificación de la fila.

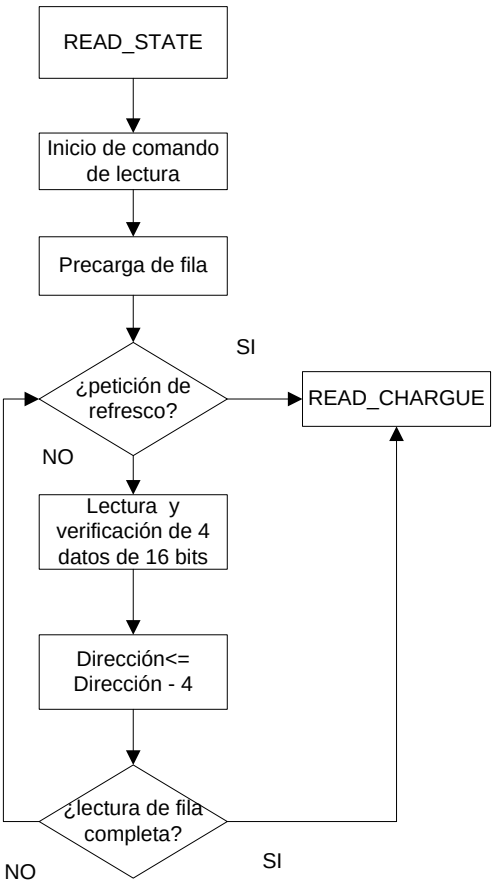

Ilustración 43: Comando de lectura de test

Se debe tener en cuenta que existe un retardo de unos 17 ciclos de reloj desde que el interfaz solicita leer en memoria hasta que el controlador proporciona los datos solicitados (y se hace la verificación de los datos).

Para la verificación de los datos, se utiliza una señal activa por nivel, de tal manera que cuando una posición de la memoria contenga un dato diferente a x"0000" o a x"FFFF" (dependiendo de la fase en la que se está ejecutando el test), se pondrá a '1', indicando que al menos un posición en memoria presenta un defecto.

Modo lectura de pila: si el sistema está leyendo la pila de la memoria, en cada comando de lectura se precargan los registros de lectura del interfaz con 2 datos de 32 bits de la memoria, es decir, se efectúa un único acceso de lectura. Para éste modo de funcionamiento no se comprueba la solicitud de comandos de refresco, debido a que al realizar un único acceso de lectura se consumen únicamente 2 ciclos de reloj (sin tener en cuenta la inicialización y finalización del comando de lectura), por lo que da tiempo a terminar la lectura y a continuación ejecutar el refresco.

El funcionamiento de este modo es idéntico al descrito en el estado read_state del modo esporádico considerándose m igual a 1.

A continuación, se comentan las transiciones del diagrama de estados.

Transiciones del diagrama de estados:

1. Una vez el controlador pone a nivel bajo los resets que actúan sobre el interfaz, automáticamente se envía el comando de inicialización al controlador.

2. Si en el estado de repose se cumple que no se estén ejecutando comandos de refresco y no halla peticiones de lectura, se iniciará automáticamente un comando de escritura pasándose a write_state.

3. Si mientras se ejecuta un comando de escritura se solicita un refresco, una petición de lectura o se completa la escritura en una fila, se pasará al estado de repose con el objetivo de ejecutar un comando de refresco o precargar una nueva fila (o banco) o bien, iniciar una lectura de la pila.

4. Cuando se recibe una petición de lectura tras un estado de escritura, se pasa a write_to_read, donde se prepara al sistema para una lectura de la pila.

5. Cuando finaliza el proceso de write_to_read se pasa a read_charge, estado en el que se gestionan los registros de lectura del interfaz.

6. Cuando se está realizando la verificación de un test_1 o no quedan datos útiles en los registros de lectura de la memoria y se recibe una petición de lectura, se pasa a read_state donde se inicia un comando de lectura en memoria.

7. Si finaliza un comando de lectura se pasa a read_chargue.

Cuando se inicia un estado de escritura de muestras en memoria tras una lectura, se pasa a repose.

Finalmente se describen las limitaciones del diseño y los puertos y bloque del interfaz.

Limitaciones del diseño:

1. Si se somete al interfaz a tests consecutivos, se debe dejar un margen de 10 ciclos de reloj entre la finalización de un test y la solicitud de inicio del siguiente.

Bloque y puertos del interfaz:

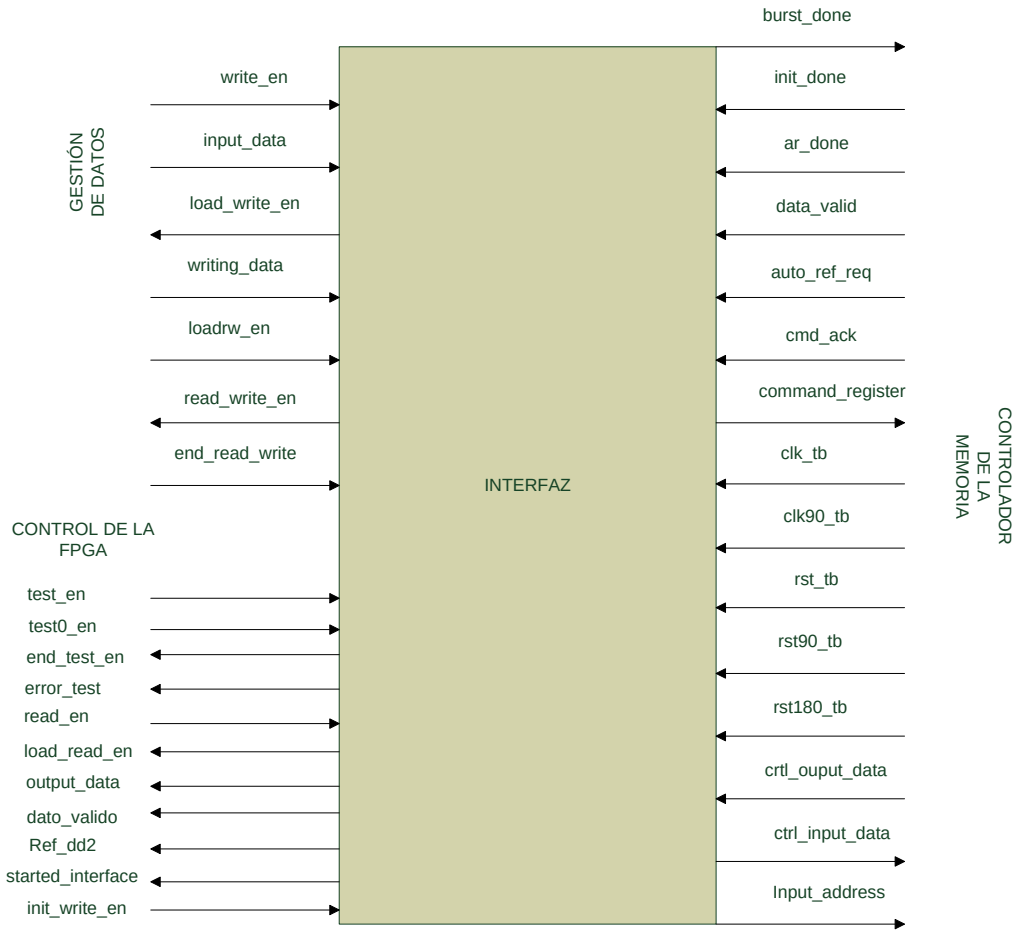

Ilustración 44: Bloque interfaz acceso continuo

Puerto	Tipo	Descripción
burst_done	Out	Esta señal se usa para terminar comandos de lectura y de escritura, poniéndose a nivel alto durante 2 ciclos de reloj (solo para un burst lenght=4). Desde el interfaz al controlador.
init_done	In	Señal activa por nivel alto, indica inicialización de la memoria completa. Desde el controlador al interfaz.
ar_done	In	Enable, indica finalización del comando de refresco desde el controlador al interfaz.
data_valid	In	Cuando está a nivel alto, indica que ctrl_output_data es válido para su lectura. Desde el controlador al interfaz.

Puerto	Tipo	Descripción
auto_ref_req	In	Petición de refresco del controlador al interfaz, aproximadamente cada 7.5 microsegundos.
cmd_ack	In	Indica acceso en memoria cuando está a nivel alto.
Command_register [2 downto 0]	Out	Comandos del interfaz para el controlador de la memoria: "000" => nada (usado en los refrescos) "010" => inicialización de la memoria "100" => escritura en memoria "110" => lectura en memoria Otros => reservado
clk_tb	In	Reloj del controlador al interfaz.
clk90_tb	In	Reloj desfasado 90 grados respecto de clk_tb.
rst_tb	In	Reset del controlador al interfaz, síncrono con el flanco de subida de clk_tb, activo por nivel alto.
rst90_tb	In	Reset del controlador al interfaz, síncrono con el flanco de subida de clk90_tb, activo por nivel alto.
rst180_tb	In	Reset del controlador al interfaz, síncrono con el flanco de bajada de clk_tb, activo por nivel alto.
Ctrl_output_data [31 downto 0]	In	Bus de lectura entre el interfaz y el controlador.
Ctrl_input_data [31 downto 0]	Out	Bus de escritura entre el interfaz y el controlador.
Input_address [24 downto 0]	Out	Bus de direccionamiento de la memoria: [24 downto 12] => Row [11 downto 2] => Column [1 downto 0] => Bank
started_interface	Out	Señal activa por nivel alto. Indica que el interfaz está inicializado tras un reset.
write_en	In	Enable. Indica que dos datos de 32 bits han sido escritos. De la gestión al interfaz.
init_write_en	In	Enable. Indica que el sistema está en estado de escritura. Del control de la FPGA al interfaz.
writing_data	In	Indica que se está escribiendo un dato de 32 bits en input_data. Activa por nivel alto. De la gestión al interfaz.
load_write_en	Out	Interfaz listo para recibir 2 datos de 32 bits de la gestión.
input_data [31 downto 0]	In	Bus de escritura entre el interfaz y la gestión.
read_en	In	Enable de petición de lectura desde el control de la FPGA al interfaz.
load_read_en	Out	Enable de dato listo para su lectura desde el interfaz al control de la FPGA.
output_data [31 downto 0]	Out	Bus de lectura entre el interfaz y el control de la FPGA.
read_write_en	Out	Enable de petición de lectura del interfaz a la gestión en el paso de escritura a lectura del sistema.

Puerto	Tipo	Descripción
end_read_write	In	Fin del paso de escritura a lectura, de la gestión al interfaz. Señal activa por nivel alto.
loadrw_en	In	Dato listo para el interfaz desde la gestión en el paso de escritura a lectura. Enable.
test_en	In	Enable de petición de test completo de la memoria. Del control de la FPGA al interfaz: 1º Escritura de x"FFFF" en la pila. 2º Verificación de la pila. 3º Escritura de x"0000" en la pila. 4º Verificación de la pila.
test0_en_n	In	Enable de petición de inicialización de la memoria. Del control de la FPGA al interfaz. Escritura de x"0000" en la pila.
end_test_en	Out	Enable de fin de test completo o de inicialización de la memoria. Del interfaz al control de la FPGA.
error_test	Out	Indica si se produce al menos un error durante la verificación. Señal activa por nivel alto. Del interfaz al control de la FPGA.

Tabla 2: Puertos interfaz acceso continuo.

2.2.4 Selección del diseño definitivo

Para la comparación entre los dos diseños realizados, se van a seguir una serie de criterios que permitan cuantificar las prestaciones de los dos diseños.

1. **Velocidad de procesamiento de datos:**

La velocidad en la lectura de los datos es la misma, de hecho, la lógica implementada es la misma. La única diferencia radica en que en el primer diseño se puede aumentar la velocidad a costa de un mayor consumo hardware. Si se tiene en cuenta que el requisito de la velocidad de lectura se cumple con un gran margen de maniobra por parte de los dos diseños, se deduce que aumentar la velocidad de lectura no es una ventaja importante.

La velocidad en la escritura de los datos cambia significativamente entre los dos diseños.

Las especificaciones del diseño requieren que el interfaz sea capaz de procesar 10 datos de 32 bits por microsegundo.

Si se tiene en cuenta que el sistema debe de ser capaz de funcionar con un reloj de 80 MHz y un reloj de 125 MHz, el peor caso posible para la velocidad de escritura de los datos, es el funcionamiento del sistema con un reloj de 80 MHz.

Los estudios realizados a continuación tienen en cuenta las condiciones más desfavorables de funcionamiento para la velocidad del interfaz. Además se debe tener en cuenta que los estudios teóricos aquí expuestos se realizan en base a la memoria

DDR2 (la diferencia del estudio en base a la DDR es despreciable). Para más información consultar capítulo 4 apartado 4.1.3 "Medición de los tiempos de acceso a la DDR2 y DDR".

Estudio de la velocidad de escritura del diseño de acceso esporádico:

Proceso	Inicio comando de escritura	n/2 accesos en memoria (n datos de 32 bits)	Fin comando de escritura	Total
Nº de ciclos de reloj	1	(n / 2) * 2 = n	28	1 + n + 28 = 29+n ciclos

Tabla 3: Análisis temporal de un comando de escritura esporádico.

En el capítulo 3, apartado 3.2 "Primer diseño escritura esporádica", se tiene una descripción completa de la variable n de la tabla anterior.

Se va a realizar el estudio en función del peor caso posible para los comandos de refresco. Las condiciones más desfavorables para un comando de refresco, es que justo antes de iniciar el comando de escritura, el controlador solicite un comando de refresco. El número de ciclos de duración de un refresco se ha obtenido mediante simulación, obteniéndose un total de 19 ciclos de reloj.

Además se considera el número de ciclos de reloj que tarda la lógica del interfaz en precargar los registros de escritura: Nº ciclos registros escritura = 2 * n.

Finalmente se puede calcular el número de datos que el interfaz puede procesar cada 7.5 microsegundos. Se ha decidido basar el estudio en 7.5 microsegundos ya que es el período con el que se ejecutan los refrescos:

Número de ciclos de reloj presentes en 7.5 microsegundos (reloj de 80 MHz): 80 * 7.5 = 600 ciclos.

Número de ciclos disponibles tras un refresco = 600 – 19 = 581 ciclos.

A continuación e obtiene el número de ciclos con acceso en memoria en un tiempo de 7.5 microsegundos:

Nº de ciclos = Nº de ciclos tras el refresco * Nº de ciclos de acceso en memoria / Total de ciclos de un comando = 581 * n / (29 + n) ciclos de acceso en memoria.

Si cada dos ciclos de reloj de acceso en memoria se escribe un dato de 32 bits, en 7.5 microsegundos, el interfaz procesa: 581 * n / (29 + n) datos de 32 bits cada 7.5 microsegundos.

Si el resultado anterior se mide en un microsegundo queda: **77.46 * n / (29 +n) datos de 32 bits procesados por microsegundo.**

Para cumplir el requisito de diseño de procesamiento en la escritura de 10 datos de 32 bits por microsegundo, se debe hallar n en la siguiente ecuación:

10 = 77.46 * n / (29 + n), se toma entonces un n = 290 / 67.46 = 4.3 = 5. **El valor mínimo de n que garantiza la escritura en memoria de 10 datos de 32 bits por microsegundo es n = 5**.

Estudio de la velocidad de escritura del diseño de acceso continuo:

Proceso	Inicio comando de escritura	256 accesos en memoria (512 datos de 32 bits)	Fin comando de escritura	Total
Nº de ciclos de reloj	1 + 3 = 4	256*2 = 512	28	4 + 512 + 28 = 544 ciclos

Tabla 4: Análisis temporal de un comando de escritura continuo

Un comando de escritura tiene una capacidad máxima de 256 accesos en memoria en direcciones diferentes (no se puede escribir en dos filas diferentes en un mismo comando). Cuando se llega a dicho límite, se debe finalizar el comando, cambiar de fila e iniciar un nuevo comando de escritura.

Los ciclos de reloj empleados en el "inicio del comando de escritura" de la tabla anterior, se deducen de: 1 ciclo de reloj desde que se solicita un comando hasta que se hace estable el comando, y 3 ciclos de reloj por despreciar el primer acceso en memoria, lo que facilita la lógica de control.

Si se tiene en cuenta que cada 7.5 microsegundos, aproximadamente, se produce un comando de refresco y si se supone el peor caso en consumo de tiempo, es decir, que la solicitud de los comandos de refresco se producen siempre en el mismo instante en el que se tiene un comando de escritura estable, se deduce el siguiente consumo de ciclos de reloj:

Duración de un refresco = fin del actual comando de escritura + finalizar el refresco + inicio del nuevo comando de escritura = 28 + 3 + 3= 34 ciclos de reloj por cada refresco.

Es importante hacer referencia a que el comando de refresco se empieza a ejecutar (y finaliza su ejecución) durante el fin del comando de escritura, por lo que el término de la suma "finalizar el refresco", en realidad, hace referencia al tiempo que tarda la lógica del interfaz en llamar a un nuevo comando de escritura. Por tanto, el término "fin del actual comando de escritura" comprende el tiempo de ejecución de un refresco y el tiempo de finalización de un comando de escritura.

Finalmente se puede calcular el número de datos que el interfaz puede procesar cada 7.5 microsegundos:

Número de ciclos de reloj presentes en 7.5 microsegundo (reloj de 80 MHz): 80 * 7.5 = 600 ciclos.

Número de ciclos tras el refresco = 600 – 34 de refresco = 566 ciclos.

Con la siguiente relación se obtiene el número de ciclos de reloj en los que hay acceso en memoria en 7.5 microsegundos:

Nº de ciclos = Nº de ciclos tras el refresco - Nº de ciclos de acceso en memoria / Total de ciclos de un comando = 566 * 512 / 544 = 532.7 ciclos de acceso en memoria.

Si se tiene en cuenta que se escribe un dato de 32 bits por cada ciclo de acceso en memoria, se tiene un total de 532.7 datos de 32 bits procesados cada 7.5 microsegundos.

O lo que es lo mismo, una **capacidad total de procesamiento de 71 datos de 32 bits por cada microsegundo con un reloj de 80 MHz.**

Por tanto, se observa que se cumple el requisito de diseño de una capacidad de procesamiento en la escritura de 10 datos de 32 bits por microsegundo.

En el capítulo 4 apartado 4.1.3 "Medición de los tiempos de acceso a la DDR2 y DDR", se utilizan técnicas de medida tanto en la simulación como en la implementación, donde se verifica la validez de los cálculos teóricos aquí expuestos.

2. Consumo hardware:

El otro criterio fundamental para la selección del diseño adecuado es el hardware implementado en cada uno de los diseños.

Para realizar el estudio de este apartado se ha hecho uso del sintetizador del software ISE de Xilinx.

Se crea un proyecto (en el directorio estudio_hardware) en el que solo se incluye el interfaz a estudiar y se utiliza la herramienta de síntesis (Synthesize – XST) e implementación (Implement Design). A continuación se consulta al sumario del diseño (Design summary) donde se expone un estudio del hardware que finalmente se implementará.

En la comparativa sobre el consumo hardware se debe tener en cuenta que los dos interfaces no presentan la misma funcionalidad:

- El interfaz de acceso esporádico contiene la gestión de datos incorporada, si bien es cierto que solo es capaz de gestionar un único bus de datos de entrada.
- El interfaz de acceso continuo precisa de un módulo de gestión de datos aparte encargado de gestionar las muestras de los módulos I/O (capaz de gestionar un número variable de buses de datos de entrada). Dicha gestión de datos se describe en el apartado 2.3. Además incorpora funcionalidades extra, como los test de la memoria, un control más robusto, etc.

Por ello se ha decidido comparar el hardware del interfaz de acceso esporádico frente al hardware del interfaz de acceso continuo más la gestión de datos requerida por el interfaz de acceso contínuo.

Estudio del hardware implementado en el diseño de acceso continuo:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	577	14752	3%
Number of Slice Flip Flops	677	29504	2%
Number of 4 input LUTs	966	29504	3%
Number of bonded IOBs	303	250	121%
Number of GCLKs	1	24	4%

Tabla 5: Estudio Hardware acceso continuo

Estudio realizado sobre una FPGA Spartan-3 1600E. La columna que contiene los valores absolutos de recursos implementados (y por tanto los valores en los que se basa el estudio) es la columna Used de la ilustración anterior.

Estudio del hardware implementado en el diseño de acceso esporádico:

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	989	11,776	8%	
Number of 4 input LUTs	1,610	11,776	13%	
Number of occupied Slices	1,090	5,888	18%	
Number of Slices containing only related logic	1,090	1,090	100%	
Number of Slices containing unrelated logic	0	1,090	0%	
Total Number of 4 input LUTs	1,716	11,776	14%	
Number used as logic	1,473			
Number used as a route-thru	106			
Number used for Dual Port RAMs	64			
Number used as Shift registers	73			

Tabla 6: Estudio hardware acceso esporádico

Estudio realizado sobre una FPGA Spartan-3 700AN. La columna que da la información de los valores absolutos (Used) es la va a servir como base para el estudio.

El estudio hardware se ha hecho tomando el valor de n = 5 calculado en el apartado de "Estudio de la velocidad de escritura del diseño de acceso esporádico".

Conclusiones del estudio y selección del diseño definitivo:

En base a los dos estudios realizados se deduce lo siguiente:

El interfaz de acceso continuo, tiene una velocidad de procesamiento de datos en la escritura 6.3 veces mayor que la velocidad del interfaz de acceso esporádico.

El hardware del interfaz de acceso continuo es drásticamente inferior al hardware del interfaz de acceso esporádico.

Number of logic utilization	**Slice flip flops**	**Total 4 input luts**	**Ocuppied slices**
Interfaz acceso esporádico	989	1716	1090
Interfaz acceso continuo	677	966	577
Relación	989 / 677 = 1.46	1716 / 966 = 1.78	1090 / 577 = 1.89

Tabla 7: Comparativa hardware interfaz

En base a los datos anteriores**, el diseño óptimo para el sistema es el interfaz de acceso continuo**.

Por tanto, el diseño del sistema, se hará teniendo en cuenta los requisitos de funcionamiento del interfaz de acceso continuo. Además de aquí en adelante, siempre que se hable de interfaz, se hará referencia única y exclusivamente al interfaz de acceso continuo.

2.3 Descripción de la gestión de los datos muestreados

En esta parte del diseño, se va a describir el bloque "gestión de los datos muestreados" para el interfaz de acceso continuo. En este bloque se realizan las siguientes operaciones:

1. Registro de las muestras provenientes de las tarjetas I/O y los registros de estado.
2. Se da formato a los datos provenientes de las muestras.
3. Se escriben los datos en el interfaz mediante un multiplexor.

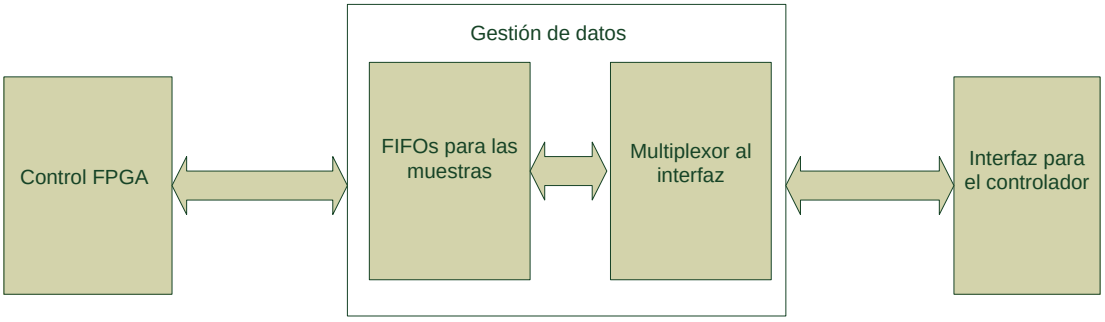

Ilustración 45: Diagrama de bloques gestión de datos

Breve descripción de las tarjetas I/O y los registros de estado:

Existen dos tipos de módulos que generan muestras, las tarjetas I/O y los registros de estado.

Una tarjeta I/O recoge datos analógicos mediante un ADC de 8 canales y datos digitales. El ADC recoge una muestra cada 1 us, y las entradas digitales se muestrean también cada 1 us. La resolución del ADC es de 13 bits. Las señales que la I/O envía de cada tipo de dato tienen 16 bits. En el caso de muestras analógicas, los 3 bits más significativos indican el canal de procedencia del ADC, el resto es el dato. En el caso de muestras digitales, son 16 bits de dato. Un registro de estado genera una muestra de 16 bits por microsegundo.

Por tanto, Como se puede ver en la siguiente tabla:

Tipo de muestra	Descripción
Muestra analógica de I/O [15 downto 0]	Los 3 bits más significativos [15 downto 13] se corresponden con el número de canal del ADC de procedencia. El valor de la muestra se corresponde con el resto de los bits [12 downto 0].
Muestra digital de I/O [15 downto 0]	Valor de la muestra digital.
Reg. Estado [15 downto 0]	Valor de la muestra.

Tabla 8: Formato de los datos en una tarjeta I/O y en un registro de estado

Formato dado a las muestras por la gestión de datos:

La escritura de muestras en la DDR establece que se escriban datos con un ancho de 16 bits. Para facilitar la lógica de control, se ha decidido que el formato dado por la gestión de datos a las muestras sea de 16 bits para el valor de la muestra y otros 16 para su clasificación (lo que suma un total de 32 bits para cada muestra). La clasificación de las muestras se hará en función del origen de las muestras, el tipo de las muestras y del tiempo en el que han sido adquiridas.

La Tabla 9 muestra el formato dado a las muestras por la gestión de datos:

Orden de los bits	Descripción
[31]	Si es una muestra procedente I/O => ‘0’. Si es una muestra procede de un Registro de Estado => ‘1’.
[30 downto 29]	Número de tarjeta I/O (los bits [30 downto 28] son utilizados para clasificar a los registros de estado).
[28]	Si es una muestra digital => ‘1’. Si es una muestra analógica => ‘0’.

Orden de los bits	Descripción
[27 downto 25]	Si es una muestra analógica => canal del ADC. En cualquier otro caso está siempre a "000".
[24 downto 16]	9 bits para la generación de la escala de tiempo de las muestras.
[15 downto 0]	Valor de las muestras. Si es una muestra analógica => [15 downto 13] extensión de signo (bit 12) [12 downto 0] valor de la muestra En cualquier otro caso => [15 downto 0] valor de la muestra

Tabla 9: Formato dado a las muestras a la salida del MUX

Como se ha explicado anteriormente, la gestión de datos se divide en un módulo de FIFOs y en el multiplexor al interfaz. En el módulo de FIFOs se van a registrar las muestras provenientes de las tarjetas I/O y los registros de estado. Además en el caso de las muestras analógicas, se va a extraer del dato recibido desde la I/O la información del canal del ADC de procedencia.

En el multiplexor al interfaz, se va proporcionar a las muestras el formato mostrado en la Tabla 9 antes de escribirlas en el interfaz. En la Ilustración 46 se muestran las etapas de la gestión de datos en que se cambia el formato de las muestras.

2.3.1 Almacenamiento de los datos en FIFOs

El bloque de las FIFOs registra las muestras provenientes de las tarjetas I/O y los registros de estado.

El módulo de FIFOs debe ser capaz de trabajar con cualquier número de tarjetas IOs y registros de estado siempre y cuando no se supere la generación de 10 muestras por microsegundo. Esta condición se deduce del requisito 2 del interfaz (apartado 2.2.1).

Por tanto, deben poder gestionarse un máximo de 4 tarjetas I/O y 2 registros de estado.

Debido a lo expuesto anteriormente, se ha hecho el diseño en base a bloques de FIFOs con capacidad para gestionar dos buses de entrada de datos. Dependiendo del número de tarjetas I/O y registros de estado presentes en el sistema, se ajustará el número de bloques de FIFOs en el diseño, de tal manera que a cada bloque de FIFOs le corresponde una tarjeta I/O o dos registros de estado. A continuación se muestra la conexión entre las tarjetas I/O, registros de estado y módulos de FIFOs:

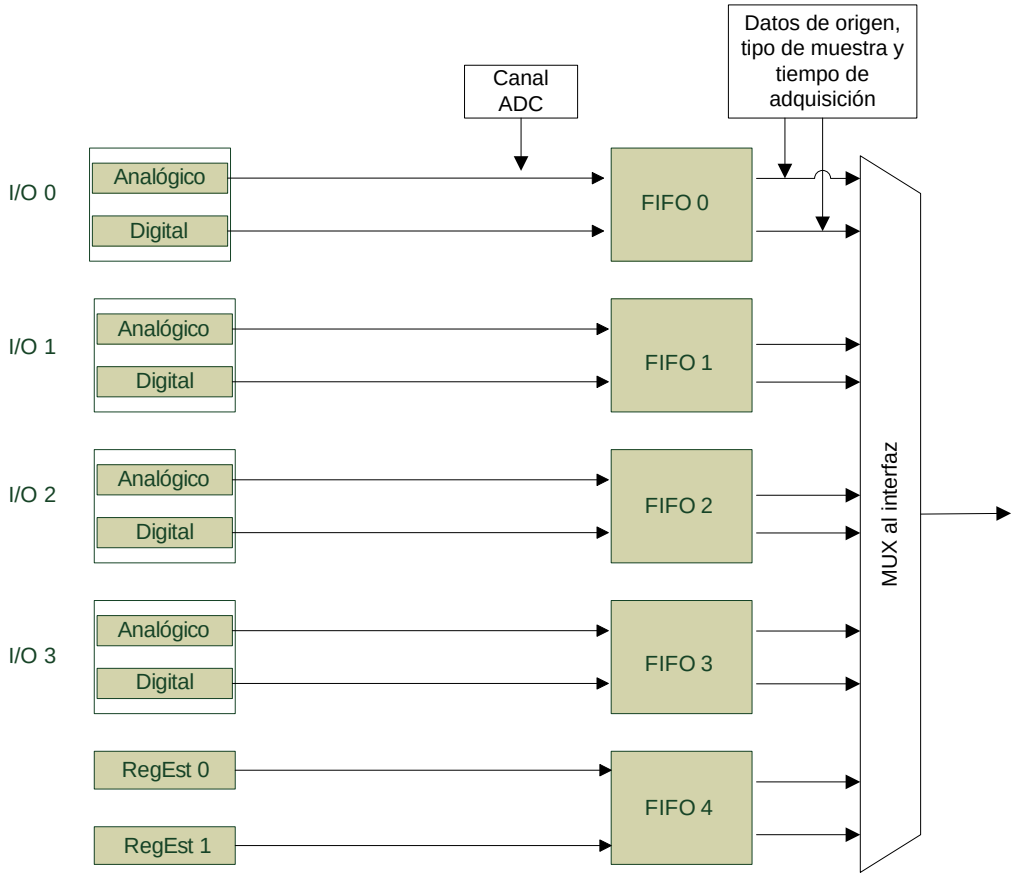

Ilustración 46: FIFOs para 4 tarjetas I/O y 2 registros de estado

Gestión de las muestras en módulos de FIFOs:

Cada módulo de FIFOs está compuesto por dos bloques de registros independientes (una para cada bus de datos de entrada). En el caso de que el módulo de FIFOs esté conectado a una tarjeta IO, uno de los bloques de registros almacenará las muestras analógicas, el otro bloque de registros almacenará las muestras digitales. Como se ha explicado anteriormente, el formato de los datos se va a dividir en un dato de 16 bits que se corresponde con el valor de la muestra y un dato de 16 bits para la clasificación de la muestra. Siguiendo esta premisa, para los datos provenientes del ADC se extraen los 3 bits que indican el canal de procedencia y se colocan en los 16 bits de clasificación (de esta manera se separa el valor de la muestra del dato de clasificación).

A continuación se muestra el formato dado a las muestras por los módulos de FIFOs:

Orden de los bits	Descripción
[18 downto 16]	En el caso de ser una muestra analógica => canal del ADC. En cualquier otro caso siempre toma valor “000”.
[15 downto 0]	En caso de ser una muestra analógica: [15 downto 13] => extensión de signo (bit 12). [12 downto 0] => se corresponde con el valor de la muestra. En cualquier otro caso: Se corresponde con el valor de la muestra.

Tabla 10: Orden de los bits en los registros de las FIFOs

Control para la lectura/escritura en los módulos de FIFOs:

Se ha diseñado una lógica para la escritura y la lectura en los módulos de FIFOs.

Por lectura o salida de la FIFO se define la lectura que ejecuta el multiplexor de las muestras almacenadas en las FIFOs.

Por escritura o entrada a la FIFO se define la escritura que ejecutan las tarjetas I/O y los registros de estado en los módulos de FIFOs.

La lógica de control utilizada en la lectura y escritura se muestra en el siguiente diagrama de flujo:

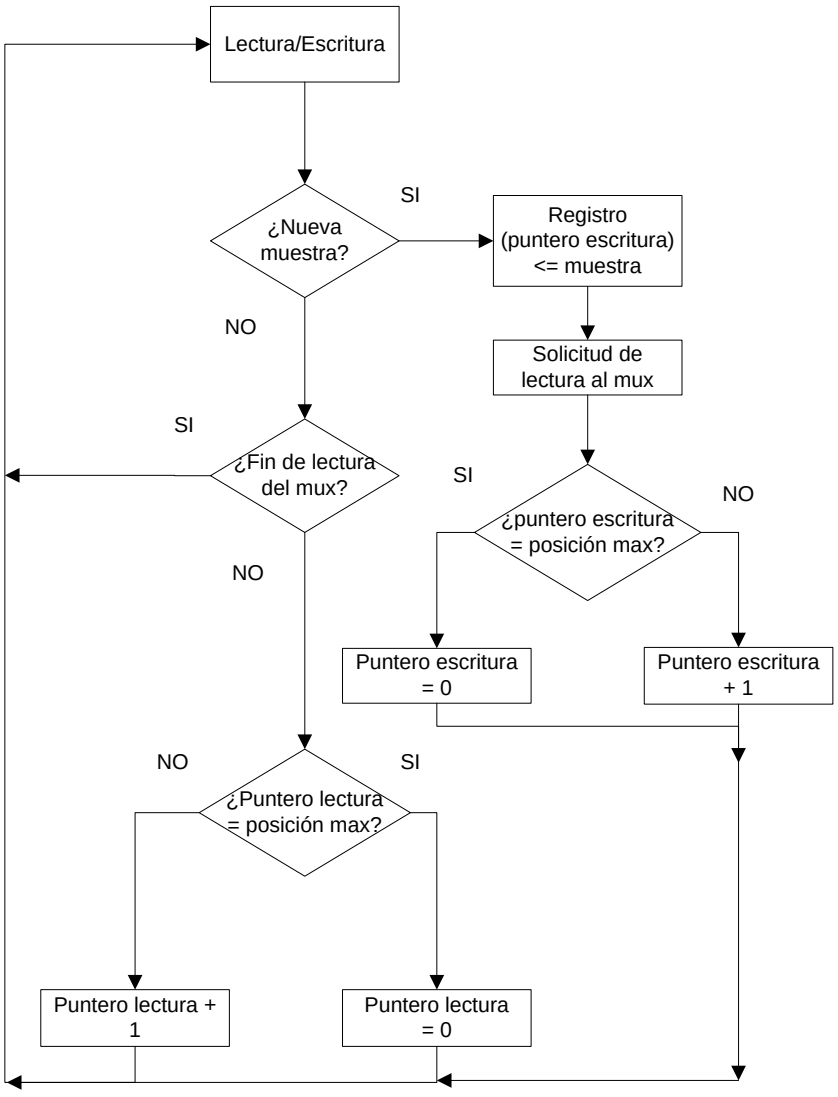

Ilustración 47: Diagrama de flujo escritura/lectura en FIFOs

El diagrama de flujo muestra únicamente la lógica de uno de los dos bloques de registros de cada módulo de FIFOs. Los punteros de lectura y de escritura son contadores que se incrementan con las señales de habilitación (enables) de "fin de lectura" y de" nueva muestra" respectivamente. Mediante los punteros se seleccionan los registros a leer o escribir.

Finalmente, es preciso proporcionar una base de tiempos a los datos. Para más información sobre la base de tiempos consultar el apartado 2.3.2 Junto con el dato se guarda información del instante temporal en el que llega y un puntero de tiempo. El puntero de tiempo, proporciona la dirección al registro que tiene almacenado el valor del tiempo en el que se almacenó la muestra.

Generación de las peticiones de lectura al multiplexor:

Por último, se describe la lógica que gestiona las peticiones de lectura al multiplexor. En el siguiente esquema se muestra el funcionamiento del módulo de FIFOs para 3 tarjetas I/O y 2 registros de estado:

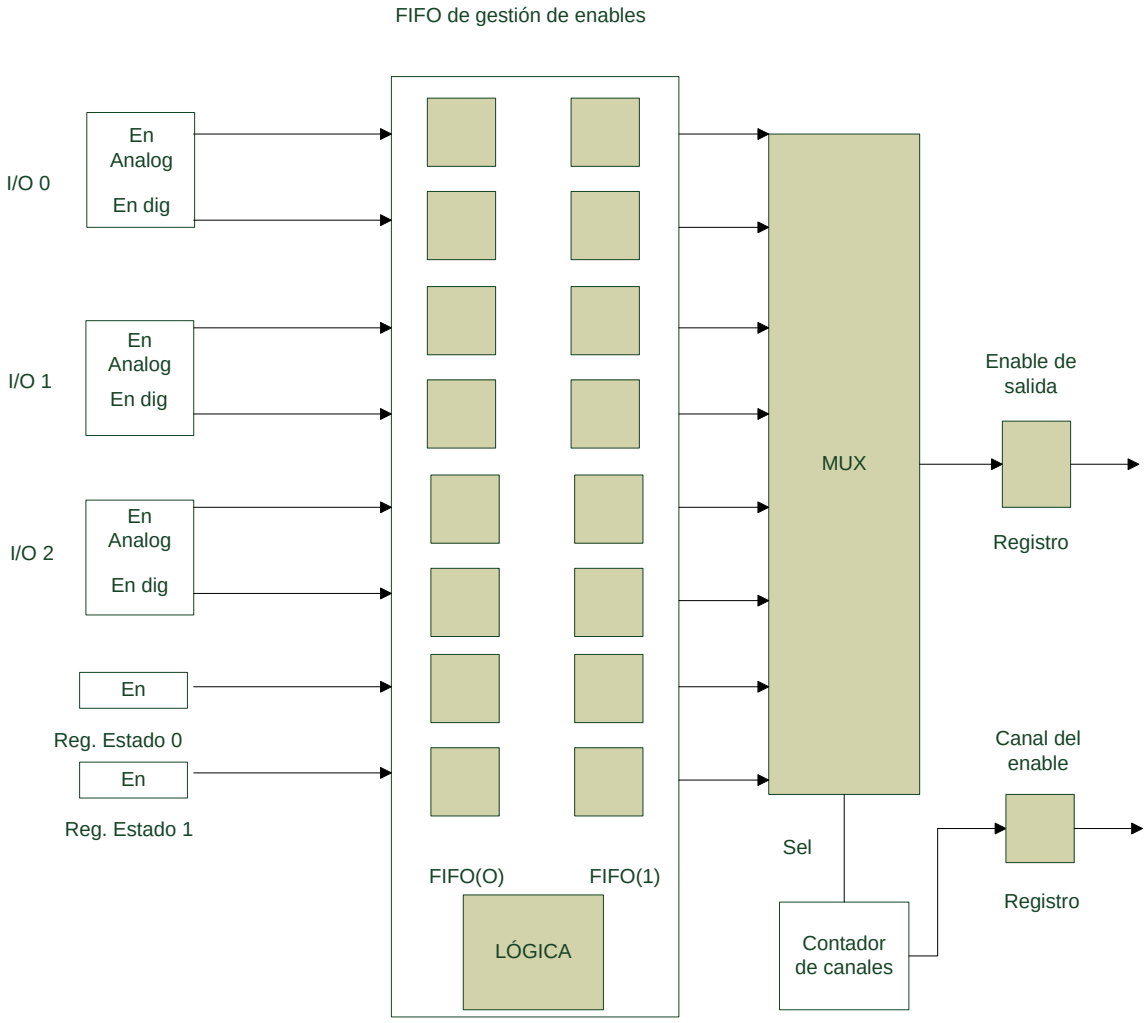

Ilustración 48: Gestión de peticiones de lectura al MUX

En el caso hipotético de que lleguen 2 ó más muestras en el mismo instante de tiempo, se debe tener una lógica capaz de gestionar todas las peticiones de lectura al multiplexor (el multiplexor no es capaz de tratar más de una petición de escritura en el mismo instante de tiempo). Las tarjetas I/O y los registros de estado, utilizan una señal de habilitación (enable) por cada bus de datos para indicar la existencia de una nueva muestra. En este sentido, se ha diseñado una lógica con la que se asegura que el multiplexor no va a recibir

más de una solicitud de lectura en el mismo instante de tiempo. Dicha lógica consiste en la implementación de un módulo de FIFOs para las habilitaciones (enables) de las tarjetas I/O y los registros de estado.

El funcionamiento de la lógica en la gestión de peticiones de lectura es simple: se basa en dos punteros, uno de escritura en la FIFO y otro de lectura de la FIFO.

El puntero de escritura indica en cuál de los dos registros de la FIFO se deben almacenar las habilitaciones (enables) de las tarjetas I/O y los registros de estado. Sólo cambia su valor cada microsegundo, pasando a apuntar al otro registro de la FIFO. De esta manera, se consigue agrupar en el mismo registro las habilitaciones (enables) que han llegado a lo largo del mismo microsegundo (es un requisito fundamental mantener el orden de tiempos de las muestras). El valor de la entrada sólo es registrado en el caso de valer '1'.

El puntero de lectura selecciona el registro de la FIFO del que se leen los datos. La lectura de los datos se realiza mediante un contador, que cíclicamente recorre todas las posiciones del registro. En el caso de que el contador seleccione una posición del registro a '1', el multiplexor (a la entrada del interfaz) recibirá la petición de lectura junto con el valor del "contador de canales". Éste último indica el canal de procedencia de la petición de lectura. Tras gestionar la petición de lectura, se pone la posición del registro a '0', para evitar que se generen peticiones de lectura de manera indefinida. El puntero de lectura cambia su valor si se cumplen las dos siguientes condiciones de forma conjunta: cada vez que transcurre un microsegundo y que no quede ninguna posición del registro direccionado por el puntero de lectura a '1'.

A continuación se adjunta el diagrama de señales de la gestión de peticiones de lectura:

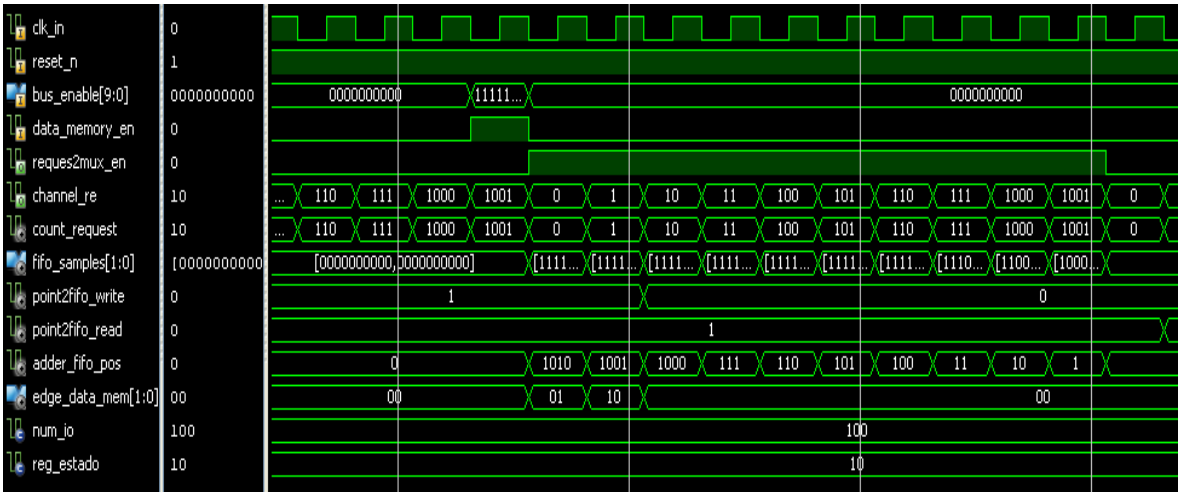

Ilustración 49: Diagrama de señales gestión de peticiones de lectura

En el diagrama de señales se muestra el funcionamiento para 4 tarjetas I/O y 2 registros de estado. Se produce la llegada de todas las habilitaciones (bus_enable) en el mismo instante de tiempo en el que transcurre un microsegundo (data_memory_en). Se observa como el puntero de escritura (point2fifo_write) se incrementa con la detección del flanco de bajada (edge_data_mem) de data_memory_en y como el puntero de lectura (point2fifo_read)

cambia de valor al finalizar la gestión de todas las peticiones de lectura en el registro 0 de la FIFO (fifo_samples).

Puertos de un módulo de FIFOs y del módulo de gestión de peticiones de lectura:

En la siguiente tabla se describen los puertos de los que consta un módulo de FIFOs.

Puerto	Tipo	Descripción
clk_in	In	Reloj de la entidad. Procedente del controlador.
reset_N	In	Reset de la entidad, activo por nivel bajo. Procedente del controlador.
data_adc [15 downto 0]	In	Bus de muestras de entrada, canal 0. Control FPGA. En caso de estar conectado a una tarjeta I/O, es el bus de muestras analógicas.
adc_en	In	Habilitación (Enable) de nueva muestra en el canal 0. Control FPGA.
data_dig [15 downto 0]	In	Bus de muestras de entrada, canal 1. Control FPGA. En caso de estar conectado a una tarjeta I/O, es el bus de muestras digitales.
dig_en	In	Habilitación (Enable) de nueva muestra en el canal 1. Control FPGA.
time_pointer	In	Puntero al tiempo de referencia de la muestra. Gestión de las muestras. El ancho de time_pointer depende del ancho de la FIFO (1 bit en la implementación).
data_write_en	In	Habilitación (Enable) de dato almacenado en la FIFO escrito por el multiplexor en el interfaz. Multiplexor.
data_time	Out	Banco de registros que almacenan los punteros de tiempo de todas las muestras almacenadas en el módulo de FIFOs. Al multiplexor.
out_data [1 downto 0] [18 downto 0]	Out	Doble bus de datos hacia el multiplexor.

Tabla 11: Puertos módulo de FIFOs

En la siguiente tabla se describen los puertos de los que consta el módulo de gestión de peticiones de lectura.

Puerto	Tipo	Descripción
clk_in	In	Reloj de la entidad. Procedente del controlador.
reset_N	In	Reset de la entidad. Procedente del controlador.
bus_enable	In	Bus de habilitaciones (enables) de las IOs y los registros de estado: 1 bit en e l bus por cada registro de estado. 2 bit en el bus por cada tarjeta I/O.
data_memory_en	In	Habilitación (Enable) que indica que el tiempo se ha incrementado un microsegundo. Del control de la FPGA.
reques2mux_en	Out	Habilitación (Enable) de petición de escritura al multiplexor.

Puerto	Tipo	Descripción
channel_re	Out	Canal de procedencia de reques2mux_en.
init_write_en	In	Esta señal indica el inicio del estado de escritura, borrándose cualquier petición de escritura anterior.
fifo_complete	Out	Indica la pérdida de habilitaciones (enables) de las I/Os y los registros de estado.

Tabla 12: Puertos módulo de gestión de peticiones de lectura

2.3.2 Generación de la base de tiempos y formato dado a las muestras

Formato dado a las muestras en el multiplexor:

Las muestras que llegan procedentes de los módulos de FIFOs al multiplexor presentan un ancho de 19 bits, en el que se especifica el canal del ADC de procedencia (en caso de ser datos analógicos) y el valor de la muestra. Para más información consultar el apartado 3.3.1 “Almacenamiento de los datos en FIFOs”.

Las muestras a la salida del multiplexor presentan un ancho de 32 bits con el siguiente orden (consultar la Tabla 9 y la Ilustración 46):

[31 downto 16] => bits de clasificación de la muestra.

[15 downto 0] => bits de valor de la muestra.

De esta forma, la clasificación de las muestras en función de los bits [31 : 28] para 2 tarjetas I/O y 2 registros de estado, es la mostrada a continuación:

Valor de los bits [31 downto 28]	Clasificación de las muestras
0000	I/O (0), canal analógico
0001	I/O (0), canal digital
0010	I/O (1), canal analógico
0011	I/O (1), canal digital
1000	Registro de estado (0)
1001	Registro de estado (1)

Tabla 13: Clasificación de las muestras para 2 tarjetas I/O y 2 Reg.Estado

Independientemente del número de tarjetas I/O, los registros de estado siempre toman el valor de “1000”y de “1001”. Por lo tanto para 4 tarjetas I/O y 2 registros de estado:

Valor de los bits [31 downto 28]	Clasificación de las muestras
0000	I/O (0), canal analógico
0001	I/O (0), canal digital
0010	I/O (1), canal analógico

Valor de los bits [31 downto 28]	Clasificación de las muestras
0011	I/O (1), canal digital
0100	I/O (2), canal analógico
0101	I/O (2), canal digital
0110	I/O (3), canal analógico
0111	I/O (3), canal digital
1000	Registro de estado (0)
1001	Registro de estado (1)

Tabla 14: Clasificación de las muestras para 4 tarjetas I/O y 2 Reg. Estado

Generación de la base de tiempos de las muestras:

El objetivo del sistema del módulo de muestreo y gestión de datos en memoria es almacenar las muestras generadas en un espacio de tiempo de 512 microsegundos. Una vez que se lea el contenido de la memoria, es fundamental poder distinguir las muestras no sólo en función de su canal de procedencia, o si es analógica o digital, sino también en función de una base de tiempos para poder establecer una relación temporal entre las muestras.

La resolución de tiempo mínima que se precisa dar a las muestras es de un microsegundo, debido a que las tarjetas I/O y los registros de estado generan una muestra cada microsegundo, mientras que la ventana máxima de tiempo que se requiere es de 512 microsegundos. Esto da un total de 512 valores diferentes para la escala de tiempos.

Por tanto se necesitan 9 bits para generar los 512 valores diferentes la base de tiempos.

El control de la FPGA envía una señal de habilitación (enable) a la gestión de los datos muestreados, indicando que el tiempo se ha incrementado en un microsegundo. Por lo tanto, es la gestión de los datos la que debe generar la base de tiempos a partir de la señal de habilitación (enable) del control de la FPGA. Debido a la rapidez de escritura del interfaz y a la forma en que se generan las muestras en las tarjetas I/O y los registros de estado, se garantiza que a lo largo de un mismo microsegundo (tiempo entre habilitaciones (enables) del control de la FPGA) el multiplexor no va a tratar muestras con más de un microsegundo de diferencia.

Si se tienen en cuenta las condiciones descritas en el párrafo anterior, se deduce que van a ser necesarios un total de 2 registros de 9 bits cada uno para guardar la base de tiempos de las muestras. El diagrama de bloques del circuito es el siguiente:

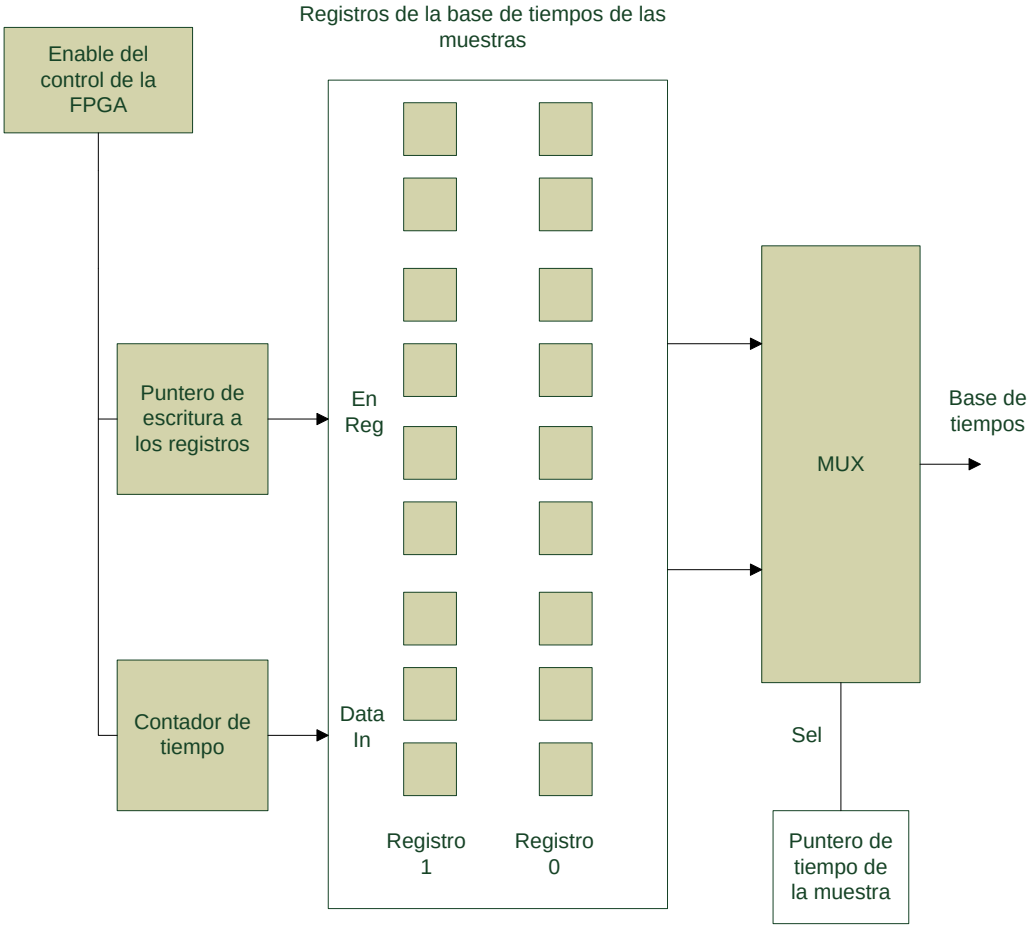

Ilustración 50: Esquema de generación de la base de tiempos

Se utiliza un contador de 9 bits para llevar la cuenta del tiempo, incrementándose cada vez que llega una habilitación (enable) de un ciclo de reloj procedente del control de la FPGA. La habilitación también incrementa el valor del puntero de escritura a los registros, de tal manera que el valor del contador de tiempo se vuelca en un registro diferente cada microsegundo. Tal y como se describe en el apartado 3.3.1 "Almacenamiento de los datos en FIFOs", cada muestra almacenada en las FIFOs cuenta con un puntero de direccionamiento al registro de tiempo correspondiente. Ese puntero de tiempo se conecta a la selección de canales de un multiplexor cuyas entradas son los registros de tiempo disponibles.

2.3.3 Multiplexación y selección de los datos a escribir en la DDR y DDR2:

Finalmente, completando la etapa "gestión de los datos muestreados", se ha diseñado un multiplexor con la funcionalidad de seleccionar la muestra que ha escribir en el interfaz. La necesidad de la inclusión del MUX, radica en que el interfaz únicamente es capaz de gestionar un único bus de escritura de 32 bits, mientras que cada tarjeta I/O genera 2 buses de escritura de 32 bits (16 bits de dato + 16 bits de información adicional) y cada registro de estado genera un bus de escritura de 32 bits (16 bits de

dato + 16 bits de información adicional). Concretamente, se selecciona la muestra correspondiente almacenada en los módulos de FIFOs, el dato de tiempo correspondiente, se indica si procede de tarjeta I/O o de registro de estado. En caso de proceder de una tarjeta I/O se indica si es una muestra analógica o digital. Por último, se indica el canal del ADC de procedencia, en caso de que la muestra sea analógica. (Consultar la Ilustración 46 y la Tabla 9).

A continuación, se muestran los bloques con los que interactúa el MUX:

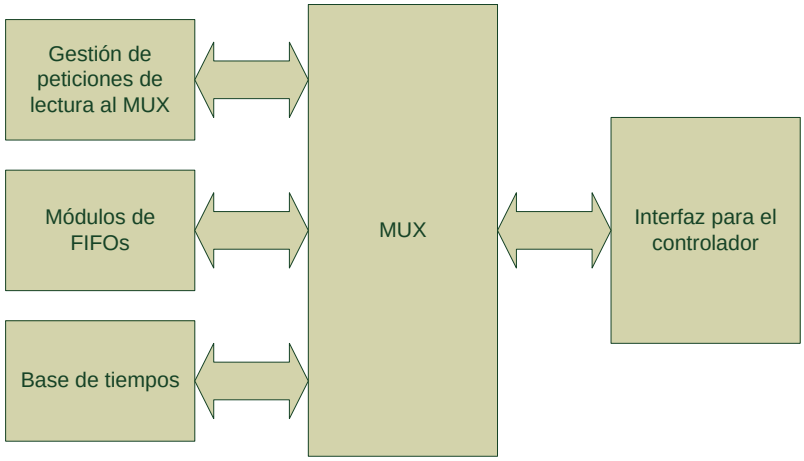

Ilustración 51: Diagrama de bloques del MUX

Recepción de peticiones de lectura:

El MUX cuenta con una FIFO encargada de seleccionar el canal de las muestras a escribir. El funcionamiento en la recepción es el siguiente: siempre que se reciba una habilitación (enable) de petición de lectura por parte del módulo "gestión de peticiones de lectura al MUX", la FIFO almacenará el valor del canal que ha solicitado la lectura, funcionando en este caso como un registro de desplazamiento, con la diferencia de que no se escribe en la entrada de la FIFO, sino en la primera posición libre.

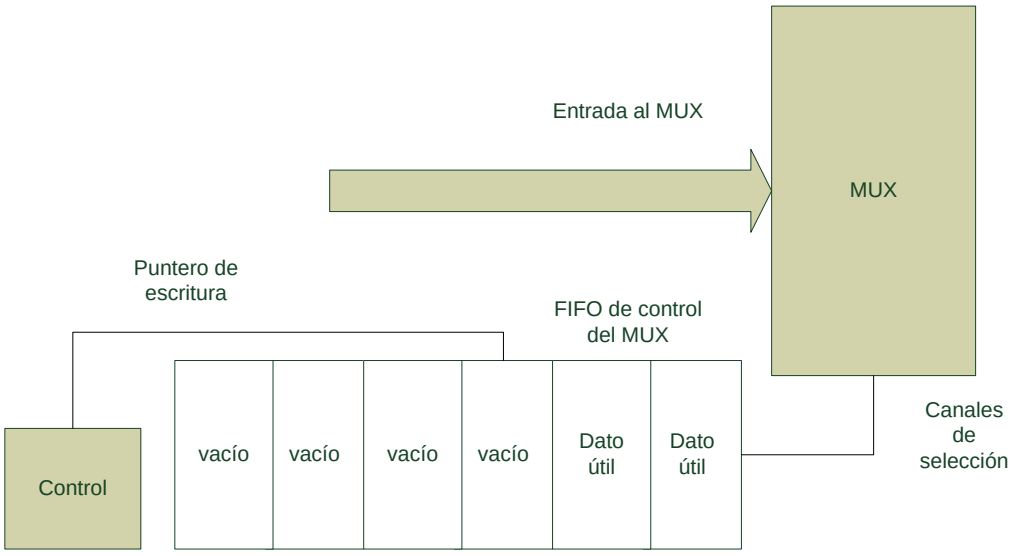

Ilustración 52: Recepción de peticiones de lectura al MUX

Dato útil: todas las peticiones de lectura provenientes de "gestión de peticiones de lectura al MUX" provocan que se almacene en la FIFO de control del MUX, el valor del canal que ha solicitado dicha petición de lectura. De tal manera que la función real de la FIFO es almacenar los canales a escribir en el interfaz.

Escritura de muestras en el interfaz:

La función del MUX no es otra que la de escribir muestras en el interfaz. Además de necesitarse un circuito que seleccione las muestras entre los canales de entrada al MUX, es necesaria una lógica para el control de los procesos de "escritura" y de "escritura a lectura" en el interfaz.

Escritura: siempre que la FIFO de control del MUX cuente con 2 datos útiles, se inicializará una escritura en el interfaz. El mínimo de dos datos útiles es debido a los requisitos especificados en el apartado 3.2.3 "Descripción del segundo diseño: acceso continuo", write_state. Cada vez que se escribe una muestra en el interfaz, la FIFO de control del MUX funciona como un registro de desplazamiento hacia la derecha, de tal manera que se posicione el siguiente dato útil de la FIFO para la selección de los canales del MUX.

Durante las operaciones de escritura en el interfaz, el MUX debe de generar la lógica de control necesaria descrita en el apartado 3.2.3 "Descripción del segundo diseño: acceso continuo", write_state. A continuación se hace una breve descripción de la comunicación MUX/Interfaz para la escritura de las muestras.

1. Cada dos ciclos de reloj el interfaz envía una habilitación (enable) al MUX indicando que está listo para la recepción de datos.

2. Siempre que el MUX tenga dos o más datos útiles en la FIFO y el interfaz le indique que está en disposición de recibir 2 muestras, se procederá a la escritura en el interfaz. El MUX escribirá una muestra en los dos ciclos de reloj inmediatamente consecutivos (por tanto un total de 2 muestras), con el formato especificado en el apartado 3.3.2 "Multiplexación y selección e los datos a escribir en la DDR y DDR2".

3. En el ciclo de reloj en el que se escriba la segunda de las muestras, el MUX indicará al interfaz que ha escrito 2 muestras mediante una señal de habilitación (enable). Además, siempre que el MUX esté escribiendo una muestra en el interfaz, lo indicará poniendo una señal a nivel alto.

4. Vuelta al punto 1.

Escritura a lectura: si el sistema pasa de un estado de escritura a un estado de lectura, existe la posibilidad de que queden datos útiles almacenados en la FIFO del MUX, por lo tanto, antes de que el interfaz lea de memoria, éste solicitará leer del MUX todas las muestras direccionadas por los datos útiles de la FIFO. En este caso el funcionamiento

del MUX será exactamente igual al descrito en "escritura" con la salvedad de que la comunicación MUX/Interfaz funcionará de la forma descrita a continuación:

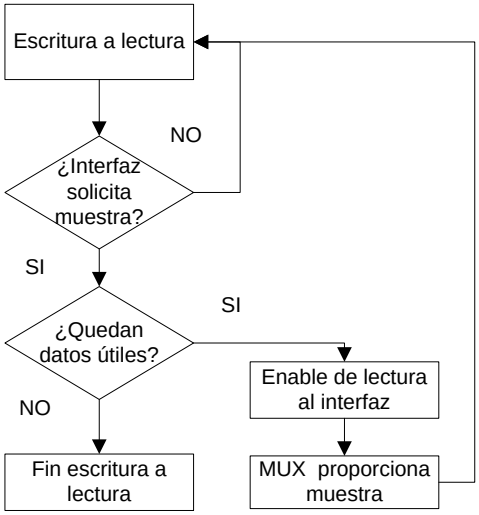

Ilustración 53: Escritura a lectura en el MUX

Se supone que cuando se pasa a lectura se dejan de recibir datos de las tarjetas I/O y los registros de estado.

Control del puntero de escritura: El puntero de escritura indica la posición de la FIFO en que debe escribirse el próximo canal que solicite una petición de lectura. La lógica que calcula el valor del puntero de escritura es la siguiente:

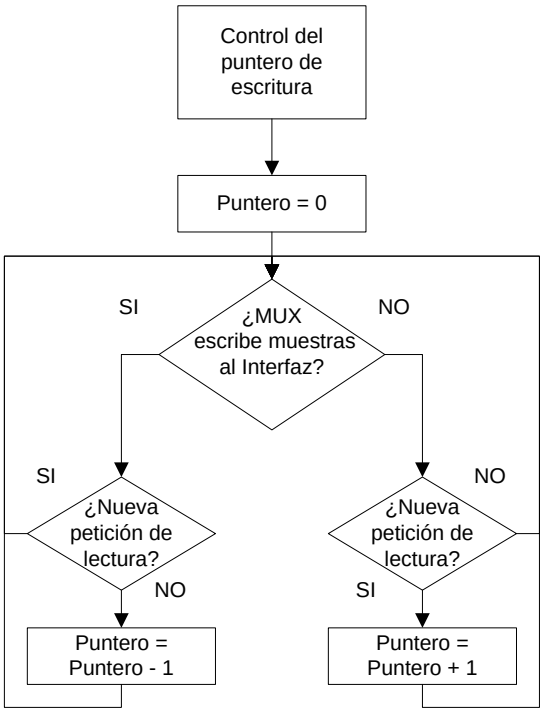

Ilustración 54: Puntero de escritura a la FIFO del MUX

Esquema general del MUX:

En el siguiente esquema se muestran las entradas entre las que debe seleccionar el MUX para 2 tarjetas I/O y 2 registros de estado:

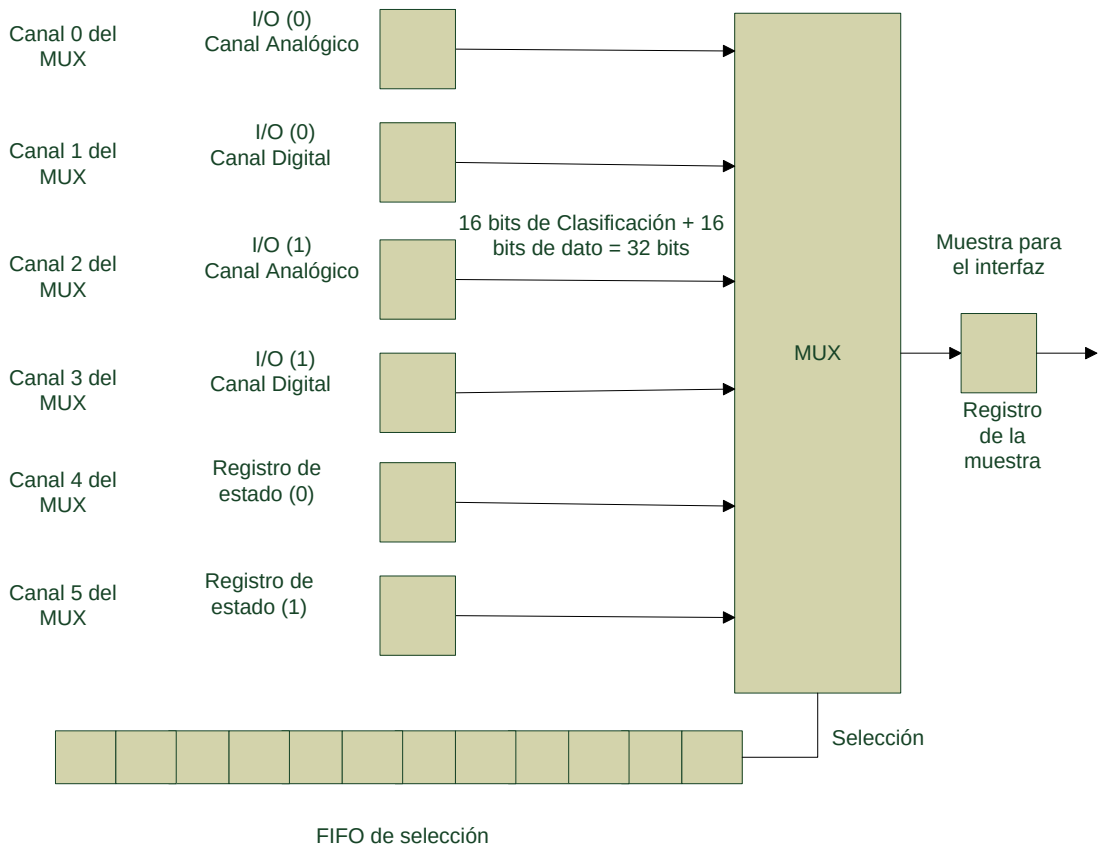

Ilustración 55: Selección de los datos en el MUX

Se recuerda que el formato de las muestras es el siguiente (de bits más significativos a menos significativos): I/O-Reg.Estado, número de tarjeta, ADC/Digital, canal del ADC, base de tiempos y valor de la muestra.

Es necesario un registro a la salida del multiplexor debido a que el tiempo de propagación de las señales entre los orígenes de los datos de las muestras y los registros de escritura del controlador de la memoria (en el interfaz no se registran las muestras) es superior o muy aproximado al periodo del reloj (dependiendo del empleo el reloj de 80 MHz o el de 125 MHz). La no inclusión de este registro podría significar una pérdida importante de muestras ya que no se podría garantizar que los registros del controlador de la memoria estén almacenando los datos deseados.

El ancho de la FIFO de selección de los canales del MUX es siempre el doble del número de canales del MUX, por la tanto un ancho de 2 * (2 * 2 + 2) = 12, para 2 tarjetas I/O y 2 registros de estado. Esto es debido a la velocidad de trabajo del interfaz y a la forma en que generan las muestras en las tarjetas I/O y los registros de estado. Si

se necesitasen más posiciones en la FIFO significaría que el sistema no es capaz de cumplir los requisitos referentes a la velocidad de escritura de muestras en memoria.

Finalmente, se incluye una lógica capaz de detectar si el MUX está perdiendo datos, debido a la falta de velocidad en el procesamiento de datos. Dicha lógica se ha diseñado de cara a verificar que el MUX cumple con las especificaciones de velocidad. El funcionamiento es simple, siempre que llegue una solicitud de lectura en el mismo instante de tiempo en el que el puntero de escritura a la FIFO direccione a la última posición de ésta, el MUX indicará la pérdida de datos.

Puertos del multiplexor al interfaz:

Puerto	**Tipo**	**Descripción**
clk_in	In	Reloj de la entidad. Procedente del controlador.
reset_N	In	Reset de la entidad, activo por nivel bajo. Procedente del controlador.
data_write_en [n -1 downto 0]	Out	Cada vez que se escribe un dato útil de la FIFO al interfaz, se envía un enable al canal de procedencia en los módulos de FIFOs, con el que se indica petición de escritura procesada. n = 2 * número de tarjetas I/O + número de registros de estado.
reques2mux_en	In	Habilitación (Enable) desde el módulo de gestión de peticiones de lectura. Indica un nuevo dato útil para la FIFO del MUX.
channel_re	In	Dato útil procedente del módulo de gestión de peticiones de lectura a almacenar en la FIFO del MUX. El valor coincide con el valor del canal que solicita la petición de lectura.
out_data [n -1 downto 0] [18 downto 0]	In	Bus de muestras de todos los módulos de FIFOs. El ancho de las muestras es de 19 bits: 3 bits más significativos. Canal del ADC en caso de ser muestra analógica. “000” en cualquier otro caso. 16 bits menos significativos. “000”+13 bits de dato en caso de muestra analógica. 16 bits de dato en cualquier otro caso. n = 2 * número de tarjetas I/O + número de registros de estado.
bus_time [n–1 downto 0]	In	Cada posición de bus_time representa un puntero a los registros de tiempo para la muestra situada en la misma posición de out_data: Bus_time (0) es el puntero de tiempo de la muestra situada en out_data(0). Procedente de los módulos de FIFOs. n = 2 * número de tarjetas I/O + número de registros de estado.
register_time [1 downto 0] [8 downto 0]	In	Registros de 9 bits que proporcionan la base de tiempos a las muestras. Son direccionados por los punteros de bus_time.

Puerto	Tipo	Descripción
load_write_en	In	El interfaz está listo para recibir 2 muestras del multiplexor (sólo en el modo de escritura). Habilitación (Enable).
write_en	Out	El MUX ha escrito dos datos en el interfaz (sólo en el modo de escritura). Habilitación (Enable).
writing_data	Out	Siempre que el MUX escriba un dato en el interfaz, writing_data = '1', en cualquier otro caso '0' (sólo en el modo de escritura).
input_data [31 downto 0]	Out	Bus de escritura en el interfaz.
end_read_write	Out	Se indica al interfaz si quedan datos útiles en la FIFO del MUX. '1' si no quedan datos útiles. '0' si quedan datos útiles.
read_write_en	In	El interfaz solicita al MUX una muestra de 32 bits. El interfaz tiene conocimiento de la existencia de datos útiles en el MUX (sólo en el modo de escritura a lectura). Habilitación (Enable).
loadrw_en	Out	Una muestra de 32 bits ha sido escrita en el interfaz (sólo en el modo escritura a lectura). Habilitación (Enable).
samples_lost	Out	Siempre que se pierda una muestra samples_lost tomará el valor de '1' hasta que llegue la próxima muestra. En el caso de que la próxima muestra también se pierda se mantendrá a '1'. En cualquier otro caso tomará el valor de '0'.

Tabla 15: Puertos del multiplexor al interfaz

2.4 Validación del módulo de muestreo y gestión de datos en memoria.

En este apartado se va a verificar el correcto funcionamiento del módulo mediante la simulación y la implementación del diseño en una FPGA. La finalidad de este apartado es comprobar la respuesta del módulo cuando es sometido a una serie de estímulos que recreen fielmente las condiciones reales de servicio. También se medirán ciertas características relativas principalmente a los tiempos de acceso a la memoria. Obteniéndose por tanto la capacidad de procesamiento de datos del módulo, el tiempo de inicialización, etc.

Como herramienta de simulación se dispone del simulador comercial del software ISE de Xilinx versión 12.3, denominado ISIM. La principal ventaja de ISIM frente a la versión libre de Modelsim es la posibilidad de simular un diseño mixto VHDL-Verilog. La importancia de esta característica radica en que el modelo de simulación de las memorias se proporciona únicamente en lenguaje Verilog, lo que obliga a utilizar un simulador capaz de simular simultáneamente tanto VHDL como Verilog.

La implementación del sistema estará condicionada al hardware existente en cada placa, esto es, gestión del reloj, uso de LEDs, switches, pines externos, etc. A continuación se expone con detalle la arquitectura para la implementación utilizada en cada caso.

Arquitectura1: kit de Xilinx Spartan-3A/3AN FPGA Starter Kit (DDR2)

El Diagrama de bloques de la arquitectura para la simulación es el siguiente:

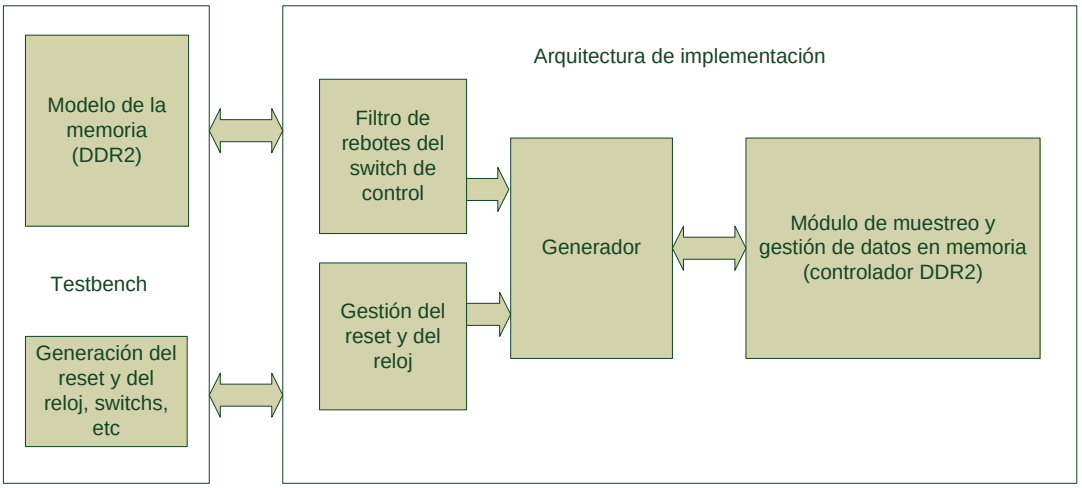

Ilustración 56: Arquitectura de implementación 1

En la impletentación se sustituirá el testbench por el hardware existente en la placa: memoria DDR2, oscilador para la generación del reloj en la FPGA, un switch de reset, un switch para el control escritura/lectura, unos pines para la medida de los tiempos de acceso y 8 LEDs para aportar información sobre el estado actual del sistema y los errores presentes en éste.

Arquitectura2: Tarjeta Spartan3 1600E (DDR)

El diagrama de bloques de la arquitectura para la simulación es el siguiente:

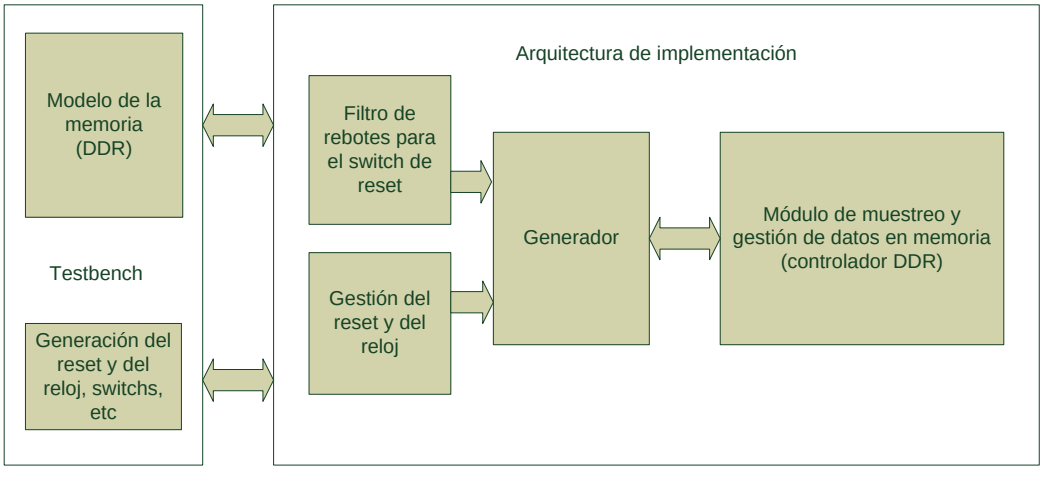

Ilustración 57: Arquitectura de la implementación 2

El hardware empleado para la arquitectura 2 es: la memoria DDR, un pulsador de reset, un LED para aportar información sobre la presencia de errores, un oscilador para la generación del reloj en la FPGA y un conector externo para posibilitar la medida de diversas señales con el osciloscopio.

2.4.1 Gestión del reset y del reloj y filtro para los rebotes de dispositivos:

El módulo de muestreo y gestión de datos en memoria precisa de 2 relojes desfasados un cuarto de ciclo entre sí con una frecuencia de 125 MHz en el caso del la memoria DDR2 y de 80 MHz en el caso de la memoria DDR. El controlador de la memoria genera todos los resets necesarios para todos los componentes de la arquitectura. Es preciso un reset global que actúe sobre el controlador y además, que sea síncrono con el reloj que presenta un desfase de 0º. Si se tiene en cuenta que la placa de la DDR2 cuenta con un oscilador de 50 MHz (entre otros) y que la tarjeta de la DDR cuenta con un oscilador de 125 MHz, es necesaria la inclusión de un módulo que genere los relojes necesarios a partir de los disponibles y que genere el reset global para el controlador.

Gestión del reset y del reloj:

Para la gestión de los relojes se ha decidido utilizar dos DCMs en cascada, uno para adaptar el reloj externo a la frecuencia de funcionamiento del sistema y el siguiente para generar dos relojes con un desfase de cuarto de ciclo a partir de la frecuencia de funcionamiento. Además, el reset de entrada al controlador será el reset generado por el segundo DCM.

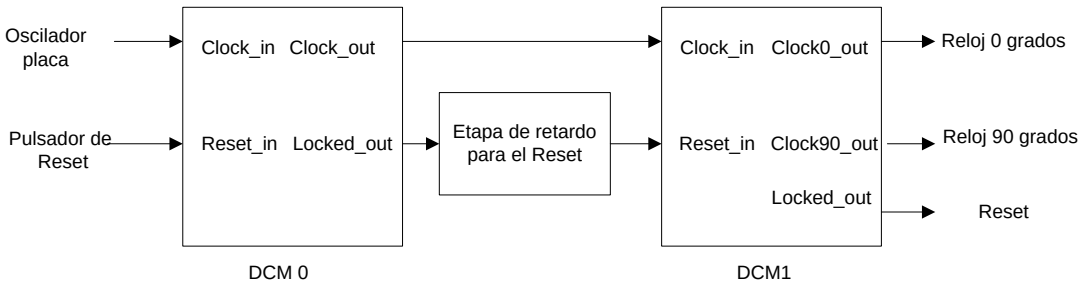

Ilustración 58: Generación del reset y del reloj para el controlador

Es necesaria una etapa de retardo para el reset entre los DCMs, consistente en un registro de desplazamiento serie de 8 biestables (los biestables tienen como reset síncrono Locked_out del DCM0). Esto evita que se deshabilite el reset sobre el DCM 1 antes de que el reloj a la salida del DCM 0 sea estable. Tanto en la DDR como en la DDR2, el esquema usado en la gestión del reloj es análogo. La diferencia radica en las frecuencias de entrada y salida de la gestión del reloj.

Para el diseño de la DDR se utiliza como reset externo el único pulsador presente en la tarjeta. Como reloj externo de entrada a los DCMs se utiliza el oscilador de la tarjeta de 125 MHz, localizado en el pin "U10" de la FPGA.

Para el diseño de la DDR2 se utiliza como reset externo el switch '1' de la placa, localizado en el pin "U10". Ver Ilustración 62 (el segundo switch empezando por la derecha). Como reloj externo se utiliza el oscilador de 50 MHz localizado en el pin "E12" de la FPGA.

Esquema global para la gestión del reset y del reloj del sistema:

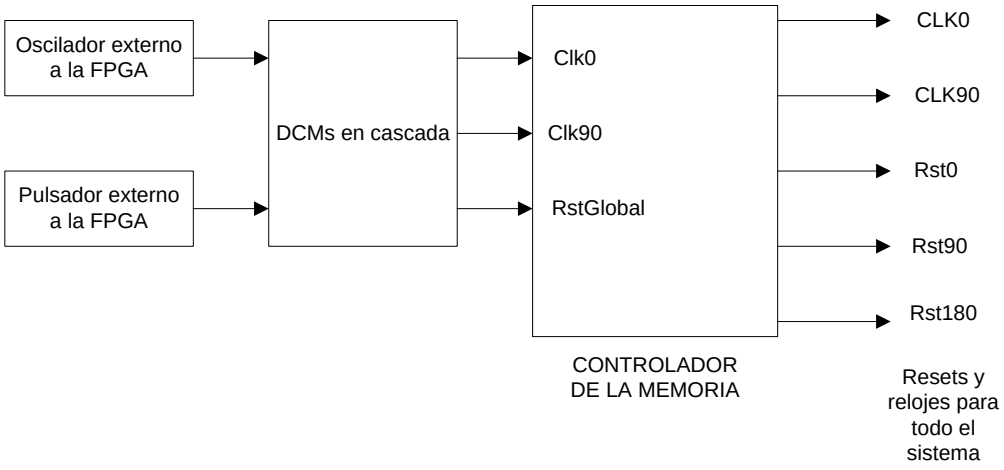

Ilustración 59: Gestión global del reset y del reloj

Filtro para los rebotes:

En la validación de la DDR y de la DDR2, se han hecho uso de switches y pulsadores. En el caso de la DDR, un pulsador para actuar como reset global del sistema. En el caso de la DDR2 un switch para controlar la generación de estímulos o el inicio de la lectura de la pila. Estos dispositivos presentan una serie de oscilaciones hasta que su salida es estable. Estas oscilaciones no suponen una amenaza para la integridad física del circuito, pero sobretodo en el caso del switch de control lectura/escritura, puede dificultar el control del sistema por parte del usuario. Siempre que el usuario quiera colocar el switch en posición de generación muestras (escritura), implicará pequeños periodos de tiempo en los la arquitectura oscila entre escritura, lectura e inicialización (misma lógica al colocar el switch en modo lectura). Para evitar estas situaciones se ha diseñado un filtro para los rebotes.

El filtro para los rebotes consiste básicamente en un temporizador que se activa ante la llega de un flanco del dispositivo (de subida o de bajada). Una una vez que transcurre un tiempo programable (5 milisegundos), se toma la salida del dispositivo como válida.

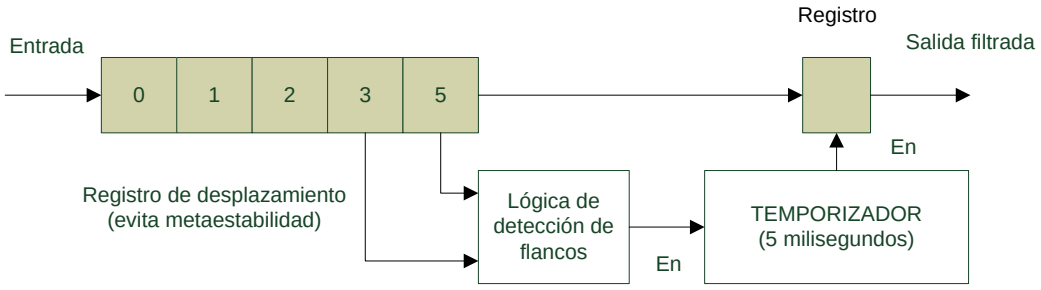

Ilustración 60: Filtro para los rebotes

2.4.2 Adaptación de los modelos de las memorias DDR y DDR2 a la simulación.

Por defecto, el MIG proporciona un testbench para comprobar la correcta funcionalidad del controlador de la memoria y del interfaz de la memoria. No se ha hecho uso de este testbench debido a que no se pretende comprobar únicamente la funcionalidad de una parte del sistema, sino que se requiere comprobar la funcionalidad del sistema al completo. Como nota adicional, el modelo de la memoria está únicamente en Verilog, lo que obliga a la simulación a trabajar con un diseño mixto.

Los modelos de memoria que proporciona Xilinx presentan ciertos problemas a la hora de adaptarse al diseño para el cual han sido generados. También existe la posibilidad de variar ciertos parámetros del modelo como pueden ser retardos, capacidad de la memoria, etc. A continuación, se exponen las correcciones que se han hecho en los modelos de las memorias.

Modelo de la memoria DDR2: (señales referidas a los puertos del modelo)

- **Corrección del ancho de los parámetros "ba" y "addr":** la memoria DDR2 posee un bus de direcciones de 25 bits dividido físicamente en dos buses de direcciones.

  Bank (ba): 2 bits que seleccionan entre 4 los posibles bancos de la memoria de 128 Megabits cada uno.

  ADDRESS (addr): bus que direcciona a la posición del banco seleccionado, alterna su valor en el tiempo entre COLUMN (10 bits de direccionamiento de la columna) y de ROW (13 bits de direccionamiento de la fila).

  El modelo proporciona por defecto un ancho de 3 bits a "ba" y un ancho de 14 bits a "addr", mientras que la memoria física con la que se está trabajando presenta unas características en esos puertos de 2 bits para "ba" y de 13 bits para "addr". Esto es debido a que el modelo de la memoria está diseñado para funcionar con varios tipos de memoria. Por lo tanto los bits más significativos de "ba" y de "addr" del modelo de la memoria se conectarán a '0' durante toda la simulación.

- **Incremento del número de posiciones de memoria:** por defecto, el modelo de simulación para la memoria DDR2 dispone de 1024 posiciones de 16 bits para la simulación. Se precisa de un número mínimo de posiciones de memoria que garantice el almacenamiento de los últimos 512 microsegundos de muestras de 4 tarjetas I/O y de 2 registros de estado.

  A continuación se calcula el número de posiciones de memoria necesarias:

  512 microsegundos * (4*2 + 2) muestras/microsegundo = 5210 muestras de 32 bits en los últimos 512 microsegundos, lo que supone un mínimo de 10420 posiciones de memoria. Para facilitar la gestión del bus de direcciones se utilizará un número de posiciones potencia de 2, lo que da un total de 16Kword de posiciones. El número de posiciones calculado también garantiza que en la simulación se utilicen varias filas de la memoria. Verificándose el funcionamiento del sistema ante los cambios de fila.

  Para cambiar este parámetro se debe acceder al paquete de datos del modelo, denominado "ddr2_model_parameters". En el paquete se busca el parámetro "MEM_BITS". MEM_BITS representa el número de posiciones de memoria medidas en Kwords. Basta cambiar el valor por defecto 10 por el número 16 para obtener 16kwords de posiciones en la simulación.

Modelo de la memoria DDR: (señales referidas a los puertos del modelo)

- **Corrección del ancho de los parámetros "ba" y "addr":** Los problemas de "ba" y ""addr" son análogos a los del modelo de la DRR2 y presentan la misma solución.

- **Incremento del número de posiciones de memoria:** Análogo al modelo de la memoria DDR2. En este caso, el parámetro que hay que modificar es "Part_mem_bits".

- **Corrección del reloj diferencial:** el reloj usado en la memoria DDR es un reloj diferencial cuyas señales toman el nombre de: ck y ck_n. El controlador tiene declaradas las dos señales del reloj diferencial como std_logic_vector(0 downto 0), mientras que el modelo tiene declaradas las señales como std_logic.

  La solución en éste caso es una conversión con una señal std_logic emplazada entre el controlador y la entrada al modelo de la memoria.

2.4.3 Máquina de estados empleada en la validación

En validación se va a emplear una máquina de estados que sea capaz de generar los estímulos y realizar las mediciones necesarias en tiempo real.

La máquina de estados debe ser sintetizable, esto es, va a ser necesaria no solo en la simulación, sino también en la implementación.

Debido a la diferencia de recursos hardware de las placas sobre las que están las memorias DDR y DDR2 se van a realizar dos máquinas de estados diferentes, una para cada placa. La lógica de generación de estímulos y de medida es esencialmente idéntica por lo que las diferencias entre las máquinas de estados son motivadas principalmente al hardware para el que han sido diseñadas.

Máquina de estados: kit de Xilinx Spartan-3A/3AN FPGA Starter Kit

Se recuerda que este módulo incluye la memoria DDR2.

La máquina de estados diseñada para esta placa está pensada para interactuar, aunque de una forma muy elemental, con un usuario humano. El usuario controla en tiempo real el inicio y finalización de la de generación de estímulos, la medida de los resultados y la realización de los test. Para que el usuario tenga conocimiento del proceso que está realizando la máquina de estados y de los posibles fallos que estén ocurriendo, se ha diseñado una sencilla codificación sobre los 8 LEDs y un conector presentes en la placa. A continuación se expone el diagrama de transiciones de la máquina de estados:

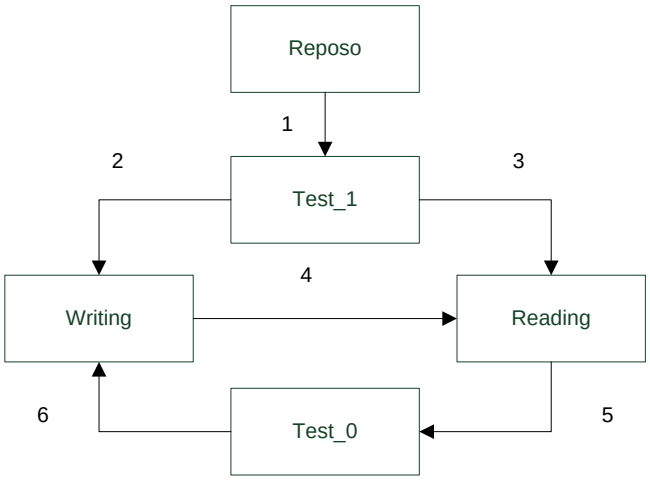

Ilustración 61: Diagrama de estados validación DDR2

Descripción de los estados estados:

Test_1 y test_0: la máquina de estados se mantiene en espera a la espera de la finalización de test correspondiente. Toda la información relativa a los test de la memoria se puede encontrar en el apartado 2.2.3.

Writing: generación de las muestras. La descripción completa se encuentra en el apartado 2.4.5.

Reading: Transcurrido un tiempo T configurable de 2 us, se inicia la medición de los resultados. La descripción completa se encuentra en el apartado 2.4.4.

Paralelamente a la máquina de estados, funciona el circuito de medida de acceso a la memoria. Información disponible en el apartado 2.4.6.

Transiciones:

1. Al finalizar el estado de reset se inicia automáticamente un test_1 sobre la memoria (verificación y borrado).
2. Al finalizar el test_1 el usuario indica a la máquina de estados, mediante el switch '0', que debe iniciar la generación de muestras.
3. Al finalizar el test completo el usuario indica a la máquina de estados , mediante el switch '0', que debe iniciar la medición de los resultados (lectura en memoria).
4. El usuario indica a la máquina de estados que debe iniciar la medición de los resultados (lectura en memoria).
5. Si el usuario decide iniciar la generación de estímulos tras el estado de medición, automáticamente se realiza un test_0 (borrado) sobre la memoria.
6. Tras finalizar el test_0 sobre la memoria se inicia la generación de muestras.

Hardware de la placa empleado:

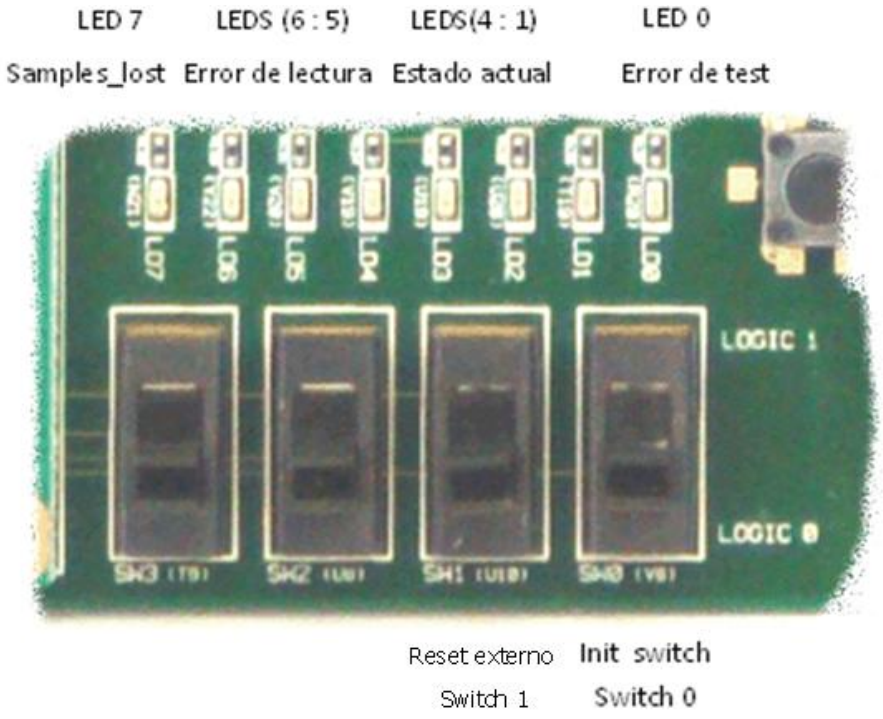

Ilustración 62: Hardware empleado en el control de la máquina de estados

Init_switch: siempre que esté a nivel lógico '1' se interpretará como una orden de inicio de generación de muestras. Cuando esté a nivel lógico '0' se interpretará como una orden de medición de los resultados. Pin "V8" de la FPGA.

Error de test: cuando el LED 0 luzca implicará que se ha detectado al menos un fallo durante la ejecución de un test completo de la memoria. Pin "R20" de la FPGA.

Samples_lost: El LED 7 lucirá de manera permanente en el caso de que se detecte la pérdida de una muestra en el multiplexor al interfaz, debido a un desbordamiento en la FIFO de selección del multiplexor. Pin "W21" de la FPGA.

Error de lectura: El LED 6 se encenderá de manera permanente en el caso de que se produzca un error en el cambio de secuencia. El LED 5 se encenderá de manera permanente en el caso de que se produzca un error en la secuencia (consultar apartado 2.4.6). Pines "Y22" y "V20" de la FPGA (del LED 6 al LED 5).

Estado actual: se corresponde con la siguiente tabla de verdad, un '1' lógico es un LED encendido, un '0' lógico es un **LED** apagado. Pines "V19", "U19", "U20" y "T19" de la FPGA (del LED 4 al LED 1).

Codificación de los bits(4 downto 1) => LEDs (4 downto 1)	Estado actual
0000	Reposo
0001	Test
0010	Test_0
0011	Writing
0100	Reading
0101	Estado inexistente

Tabla 16: Codificación de los estados de validación DDR2

Se ha utilizado el conector J18 de la placa para la medida de los tiempos de acceso a la DDR2. (Imagen obtenida del documento Spartan-3A/3AFPGA Starter Kit Board User Guide" UG334 de Xilinx)

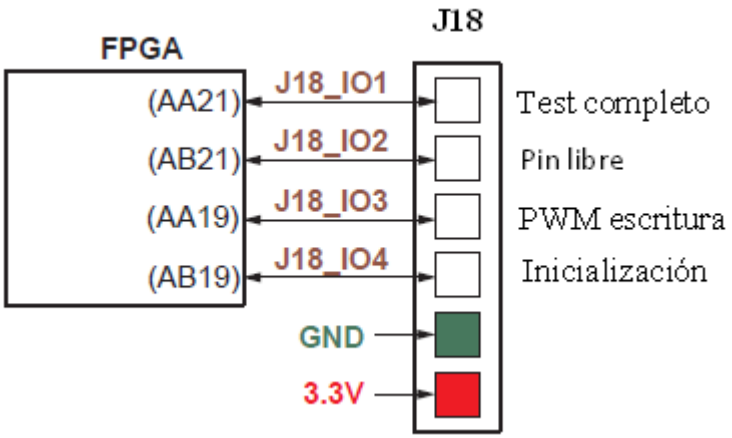

Ilustración 63: Conexionado físico para la medida de los tiempos de acceso

Test completo: Señal para la medida del tiempo de realización de un test completo.

PWM escritura: onda PWM para medir el acceso a memoria durante la escritura de muestras.

Inicialización: Señal para la medida del tiempo de inicialización de la memoria (test_0).

Máquina de estados: Tarjeta Spartan3 1600E

Se recuerda que este módulo incluye una memoria DDR. A diferencia de la anterior máquina de estados, ésta está diseñada para funcionar de una forma completamente automática. Esto es debido a la inexistencia del hardware necesario para controlar la memoria (pulsadores, interruptores, etc.). Consultar la Ilustración 66.

Diagrama de estados (máquina de estados general):

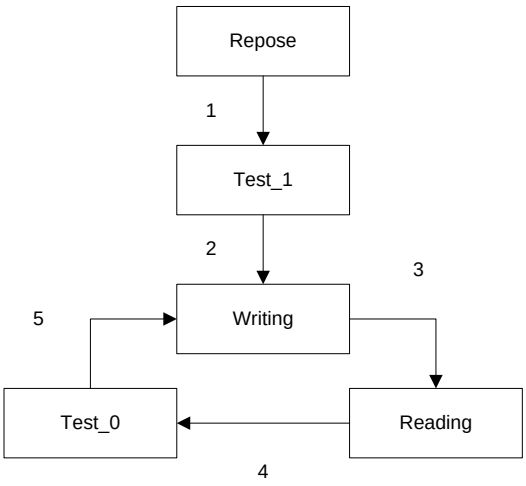

Ilustración 64: Diagrama de estados global para la validación de la DDR

La funcionalidad de cada estado es análoga a la descrita en la máquina de estados anterior.

Transiciones:

1. Al finalizar el estado de reset se inicia automáticamente un test_1 (verificación y borrado) sobre la memoria tras.
2. Tras finalizar el test_1 se inicia la generación de muestras.
3. Tras transcurrir un tiempo T_WRITE (1.25 milisegundos) desde el inicio de la generación de muestras, se inicia la medida de los resultados.
4. Una vez que se finaliza la lectura de toda la pila de la memoria, se realiza el test_0 (borrado) sobre la memoria.

5. Finalizada la inicialización de la memoria se vuelve a iniciar la generación de muestras. Nótese que la máquina de estados transita de manera cíclica entre los estados: writing - reading – test_0.

Se ha diseñado una máquina de estados adicional en el que se somete al módulo a un test_1 de forma cíclica (máquina de estados específica):

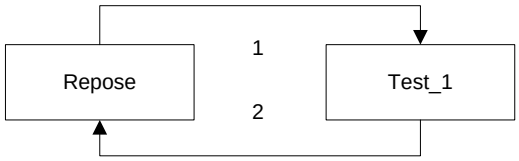

Ilustración 65: Diagrama de estados específico para la validación de la DDR

La funcionalidad es simple: siempre que transcurra un tiempo de espera (2 microsegundos) en Repose, se pasará a Test. Una vez finalizado el test completo, se volverá a l estado de espera Repose.

El hardware utilizado es el único LED disponible en la tarjeta y uno de los 4 conectores externos a la tarjeta, el "J1". A continuación se puede observar la tarjeta "Spartan-3E FPGA Industrial Micromodule" montada sobre el "Prototyping Carrier Board for Industrial Micromodule":

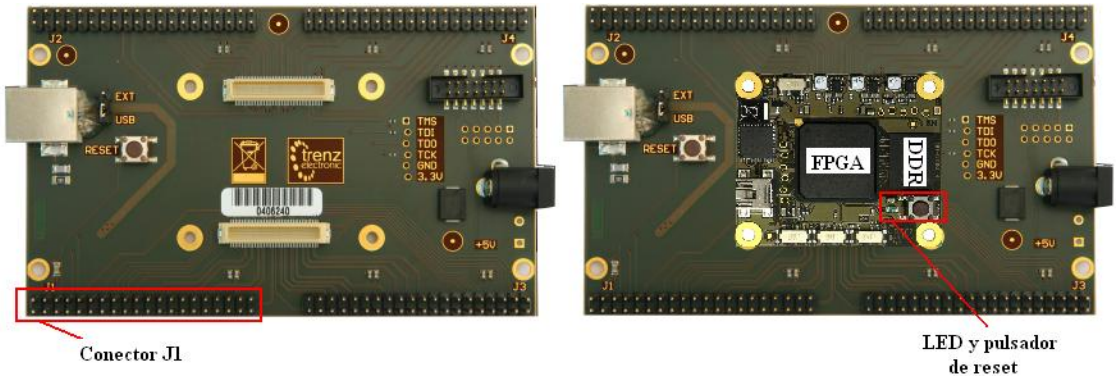

Ilustración 66: Hardware para la implementación. DDR

En el caso de que se utilice la máquina de estados general, el LED se apagará en el caso de que se produzca uno de los siguientes errores: en la validación de la lectura, en la ejecución de un test completo, por la pérdida de muestras en el multiplexor o en el caso de que la máquina de estados permanezca en el mismo estado durante un tiempo superior a 5 milisegundos.

En el caso de que se utilice la máquina de estados alternativa, el LED lucirá en caso de que se produzca un fallo durante la ejecución de los test, en cualquier otro caso parpadeará cada 0.5 segundos.

Para la medida de los tiempos de acceso a la DDR y de ciertas señales de error se ha utilizado el conector J1 de la tarjeta.

Pin conector	Nombre de señal	Pin FPGA	Tipo
35	3.3 V	-	-
11	GND	-	-
9	FIFO_complete	P1	In/Out
25	Time_test	R3	In/Out
17	Time_tes0	V7	In/Out
21	PWM	P4	In/Out
29	Ref_time1	V6	In/Out
33	Ref_time2	P6	In/Out

Tabla 17: Conector J1 tarjeta Spartan3 - 1600E

Puertos de la máquina de estados:

Puerto DDR	Puerto DDR2	Tipo	Descripción
clk_in	clk_in	In	Reloj de la entidad, desde el controlador.
reset	reset	In	Reset de la entidad, desde el controlador.
data_memory_en_n	data_memory_en_n	Out	Habilitación (Enable) que indica el incremento del tiempo en un microsegundo. Utilizado para generar la base de tiempos en las muestras.
data_in	data_in	Out	Bus de muestras generadas.
bus_enable	bus_enable	Out	Bus de habilitaciones (Enables) de las muestras.
init_write_en_n	init_write_en_n	Out	Habilitación (Enable) que indica el inicio de la escritura de muestras en memoria.
read_en_n	read_en_n	Out	Petición de lectura de un dato de 16 bits.
pwm	pwm	Out	Mide el acceso a memoria durante la escritura
time_test	time_test	Out	Mide el tiempo de ejecución de un test completo de la memoria.
time_test0	time_test0	Out	Mide el tiempo de realización de la inicialización de la memoria.
write_gen_en	write_gen_en	In	2 datos de 32 bits han sido escritos en memoria (usado en pwm). Habilitación (Enable).
load_gen_en	load_gen_en	In	Interfaz listo para procesar 2 datos de 32 bits (usado en pwm). Habilitación (Enable).
load_read_en_n	load_read_en_n	In	Dato listo para su lectura.
output_data	output_data	In	Bus de lectura de datos en memoria.
samples_lost	samples_lost	In	Pérdida de muestras en el MUX.

Puerto DDR	Puerto DDR2	Tipo	Descripción
ref_req_gen	-	In	Solicitud de refresco del controlador.
ar_done_gen	-	In	Fin del comando de refresco
command_gen	-	In	Comandos de la memoria.
ref_time1	-	Out	Duración de un comando de refresco.
ref_time2	-	Out	Tiempo entre la solicitud y la finalización de un comando de refresco.
test_en_n	test_en_n	Out	Petición de test completo. Habilitación (Enable).
end_test_en_n	end_test_en_n	In	Habilitación (Enable) de fin de test/inicialización.
test0_en_n	test0_en_n	Out	Petición de inicialización. Habilitación (Enable).
error_test	error_test	In	Error durante la ejecución de un test.
state_led	-	Out	LED de estado del sistema.
-	target_leds	Out	Estado actual/errores del sistema.
-	init_check_memory	In	'1'. Usuario da orden de generación de muestras. '0' Usuario da orden de lectura en memoria.

Tabla 18: Puertos máquina de estados para la validación

2.4.4 Generación de datos.

Para poder comprobar el funcionamiento del módulo muestreo y gestión de datos en memoria, es necesario someter a éste a una serie de estímulos. Dichos estímulos son generados por un circuito que imite el funcionamiento de un número variable de tarjetas I/O y de registros de estado.

A continuación se hace un breve recordatorio sobre el funcionamiento de los componentes y procesos a modelar:

Una **tarjeta I/O** consta de dos buses de datos de 16 bits cada uno. Cada bus de datos genera una muestra cada microsegundo (mediante una señal de habilitación se indica la existencia de una nueva muestra). Uno de los buses es de datos digitales en cuyo caso los 16 bits del bus representan el valor de la muestra. El otro bus es de datos analógicos, siendo los 3 bits más significativos coincidentes con el valor del canal de ADC del que proviene la muestra y los restantes 13 bits se corresponden con el valor de la muestra.

Un **registro de estado** consta de un solo bus de datos de 16 bits. Genera una muestra cada microsegundo (mediante una señal de habilitación se indica la existencia de una nueva muestra). En este caso, los 16 bits se corresponden con el valor de la muestra.

Existe la probabilidad de que el control de la FPGA decida que alguna muestra no es válida y por lo tanto no la envíe al módulo de muestreo y gestión de datos en memoria. En el generador de datos debe modelar esta característica.

También se debe simular la habilitación (enable) que se recibe del control de la FPGA, indicándose a la base de tiempos que el tiempo se ha incrementado en un microsegundo.

También se va a contemplar el peor caso posible para la generación de muestras. Consiste en la generación de todas las muestras en el mismo ciclo de reloj y un ciclo de reloj antes de que se reciba la habilitación (enable) de incremento de tiempo en un microsegundo. Esto permite chequear una gran parte de la funcionalidad de los circuitos de control. Esto obliga al interfaz y al MUX a generar escrituras consecutivas en memoria y obliga a la "gestión de peticiones de lectura al MUX" a tratar peticiones de lectura generadas en microsegundos diferentes.

El circuito para generar los datos está formado por 2 contadores, uno de 3 bits para generar el valor de los canales del ADC en las muestras analógicas de las tarjetas I/O y otro de 16 bits para generar el valor de los registros de estado. En el caso de muestras analógicas y digitales, solo se emplearán los 13 bits menos significativos del contador.

Se incluye un tercer contador funcionando como temporizador, con dos utilidades: Generar las habilitaciones (enables) de las muestras y la habilitación (enable) que indica el incremento del tiempo en un microsegundo.

Finalmente se incluye la lógica que simula muestras no validas para un número variable de tarjetas I/O. Está formada por un contador que indica el canal/es de las tarjetas I/O que no van a generar muestras para un mismo microsegundo. A continuación se incluye el diagrama de bloques para la generación de las muestras.

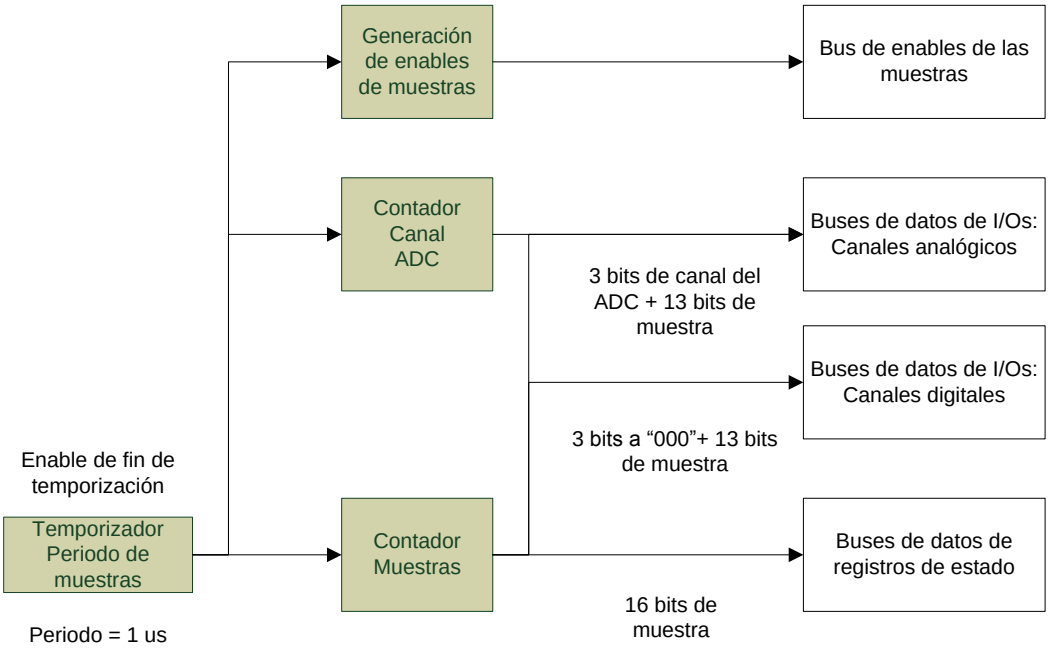

Ilustración 67: Esquema generación de datos

2.4.5 Comprobación de los datos almacenados en la DDR y DDR2.

Someter al sistema a una serie de estímulos sin un módulo capaz de medir y evaluar los resultados, no aporta una información significativa con la que poder validar el diseño. Este hecho motiva la creación de un circuito con la finalidad de comprobar el funcionamiento del circuito ante los estímulos generados por el circuito descrito en el apartado anterior, 2.4.4. La comprobación de los datos se describe mediante el siguiente diagrama de flujo:

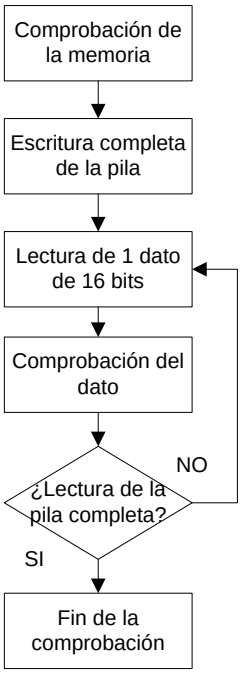

Ilustración 68: Lógica de comprobación de la memoria

La comprobación de los datos se va a basar en la medición de la base de tiempos de los datos.

La comprobación de 2 características de la base de tiempos de los datos aporta la información suficiente como para saber si el algoritmo de escritura en memoria ha funcionado de una manera correcta.

Primera característica: se va a comprobar el número de datos consecutivos que presentan el mismo valor de tiempo. Se recuerda que todas las muestras generadas en el mismo microsegundo por las tarjetas I/O y los registros de estado van a presentar el mismo valor de tiempo. Además se sabe que por cada microsegundo de muestreo cada tarjeta I/O genera dos muestras y cada registro de estado genera una muestra. Por tanto, se va a comprobar que cada n datos consecutivos se mantenga el mismo valor de tiempo. Donde n es igual a:

n = 2 * (Número de tarjetas I/O) + Número de registros de estado.

Para implementar de una manera correcta la primera característica se debe tener en cuenta el formato de los datos de lectura. Los datos de traza escritos en memoria presentan un ancho de 32 bits, los 16 bits más significativos de clasificación (contienen entre otros el dato de tiempo) y los 16 bits menos significativos el valor del dato (cada dato de traza ocupa dos posiciones de memoria). La lectura proporciona datos DDR de 16 bits, es decir, que por cada dos lecturas solicitadas al sistema se van a leer las dos partes constituyentes de uno de los datos de traza escritos en memoria, con un estricto orden de lectura de los bits de clasificación primero y los bits de valor del dato después.

Finalmente se deduce que la primera característica medida, consiste en que se mantenga el mismo valor de tiempo durante 2 * n = K lecturas consecutivas, midiéndose el tiempo únicamente en los datos DDR pares a partir del primer dato DDDR n= 0. A continuación se expone un esquema donde se aclara la lógica empleada suponiendo que únicamente se dispone de una tarjeta I/O:

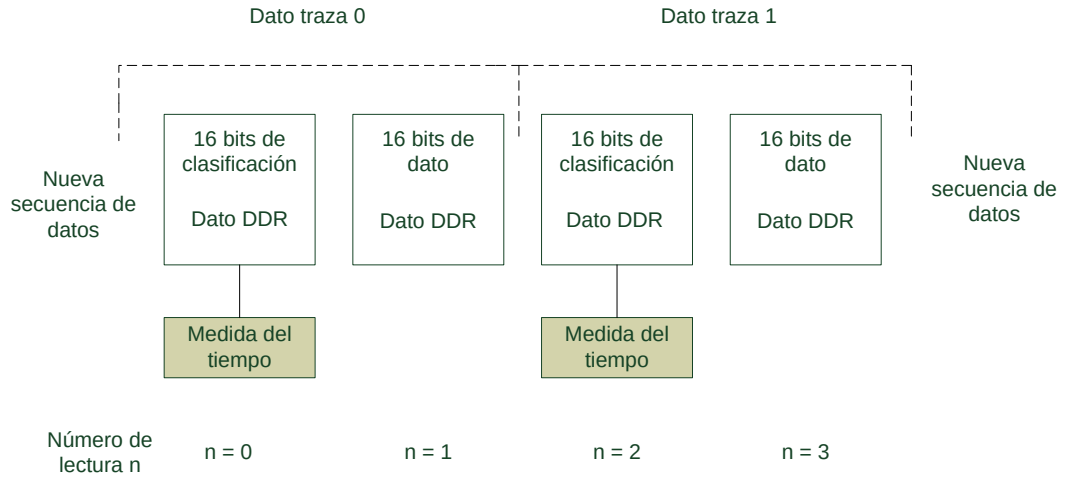

Ilustración 69: Esquema primera medida de tiempo

Si se da el caso de que una de las tarjetas I/O está averiada, es decir, no genera muestras durante la escritura de la memoria, se debe tener en cuenta a la hora de medir la primera característica. La diferencia radica en que cambiará de valor del número de lecturas (parámetro K):

K = 2 * [2 * (Núm. de IOs) + Núm. de Reg.Estado – Núm. canales de I/O averiados].

Segunda característica: Se medirá el incremento del tiempo resultante de dos secuencias de datos consecutivas.

Si se tiene en cuenta que el dato de tiempo está contenido en 9 de los 16 bits de clasificación de una muestra y que el valor del tiempo está generado por un contador cíclico que cuenta entre 0 y 511 de forma ascendente en intervalos de 1 (apartado 2.3.2), se deduce que cada n lecturas de datos, el tiempo se debe reducir en 1 unidad o en el caso de un desbordamiento en el dato de tiempo, tomar el valor 511 tras una

secuencia de n datos con un valor a 0. El parámetro K es el mismo parámetro K calculado en la primera característica.

Implementación de la máquina de estados para la comprobación de los datos:

Para la medición de las dos características citadas se ha diseñado una sencilla máquina de estados, a continuación se expone el diagrama de estados:

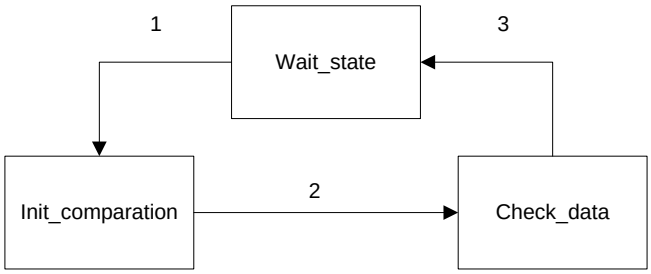

Ilustración 70: Diagrama de estados Comprobación en la lectura

Descripción de los estados:

Wait_state: es el estado de reposo, se inicializan los contadores y registros empleados en la comprobación de los datos.

Init_comparation: la función de este estado es desechar la primera secuencia de datos con el mismo valor de tiempo, debido a que existe la probabilidad de que no se hayan escrito en memoria todas las muestras correspondientes al último microsegundo de escritura (hipótesis válida en el caso de que se utilice un generador de datos para la memoria distinto al descrito en el presente proyecto o que se utilicen las propias tarjetas I/O y registros de estado). No desechar la primera secuencia de datos, puede redundar en la detección de errores en la lectura por parte de la máquina de estados, ante un funcionamiento correcto del módulo de "muestreo y gestión de datos en memoria".

Check_data: En este estado se comprueban las características 1 y 2 del tiempo. Básicamente consiste en 2 contadores. Uno de los contadores lleva la cuenta del número de datos leídos. En el momento en el que se lee la pila completa finaliza la comprobación de los datos. El otro contador lleva la cuenta del número de lecturas que se deben realizar en cada secuencia de datos. Cada dos lecturas se comprueba que el tiempo no varíe y cada K lecturas (cada secuencia de datos) se comprueba que el tiempo se decremente en 1 ó desborde de 0 a 511. Para poder comparar el tiempo entre los diferentes datos se utiliza un registro de tiempo. El registro almacena el valor de tiempo del último dato verificado. Restando '1' a la salida del registro, se puede disponer del valor del tiempo del último dato chaqueado y del valor del tiempo teórico inmediatamente posterior. A continuación se expone el diagrama de flujo que explica la comprobación de las características 1 y 2 del tiempo.

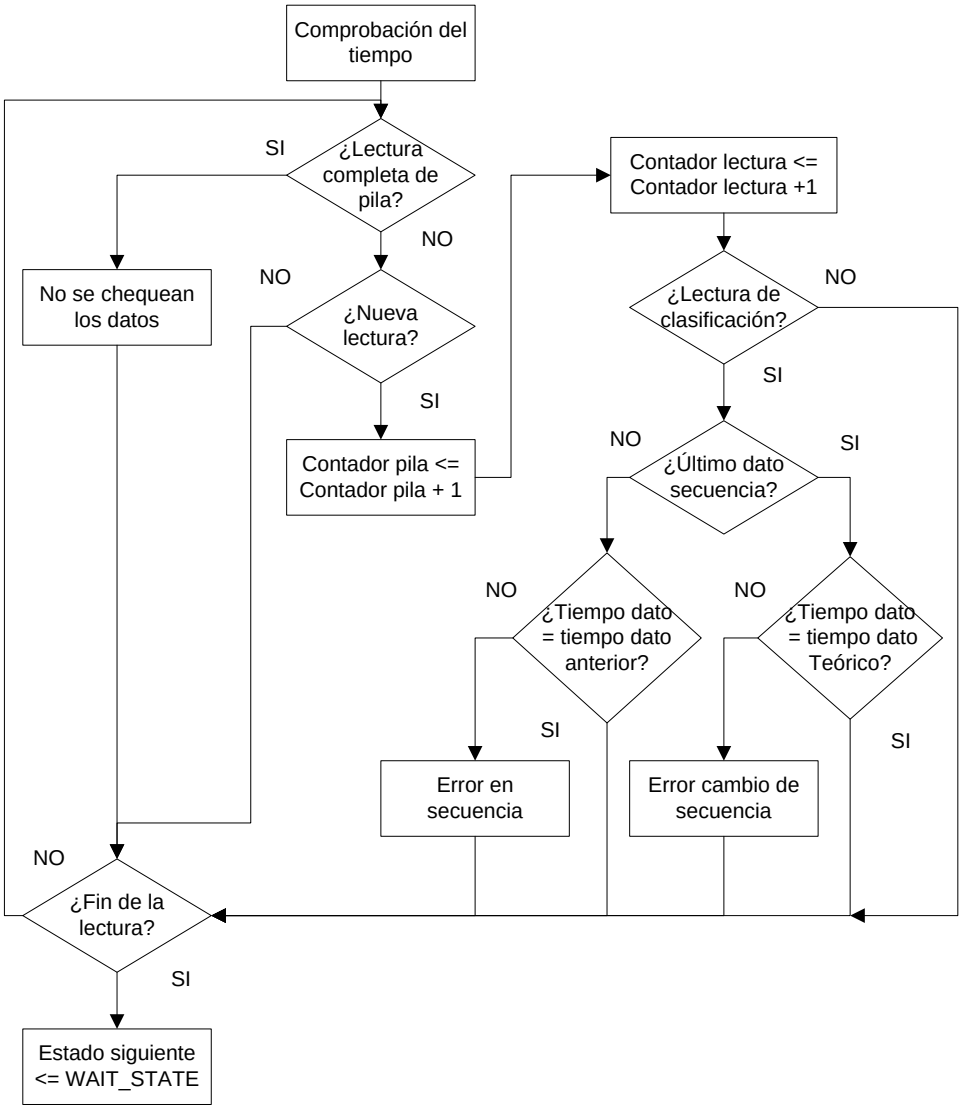

Ilustración 71: Diagrama de flujo comprobación de tiempos

A continuación, se describen las transiciones de la máquina de estados de la Ilustración 70.

Transiciones:

1. En el instante de tiempo en el que se lea el primer dato de la pila la máquina de estados iniciará la comprobación de la lectura (init_comparation).

2. Una vez finaliza la lectura de la primera secuencia de datos se pasa a la comprobación de datos (check_data).

3. Cuando se han leído todos los datos de la pila finaliza la comprobación de datos.

Conclusiones:

Mediante la lógica descrita en este apartado, se ha verificado que el módulo de muestreo y gestión de datos en memoria funciona acorde a las especificaciones de diseño.

2.4.6 Medición de los tiempos de acceso a la DDR y DDR2.

En el 2.2.4 se hizo un estudio teórico relativo al número de datos por microsegundo que se podían escribir en memoria. Para validar el estudio teórico se ha diseñado un circuito que mida las siguientes características del módulo al completo: tiempo de ejecución de un test completo, tiempo de ejecución de una inicialización (test_1 y test_0, apartado 3.2.3) y acceso a memoria durante el estado de escritura de muestras en memoria.

La medición de estos parámetros posibilita conocer realmente la capacidad de procesamiento de muestras del módulo, lo que redunda directamente en el número de tarjetas I/O y de registros de estado que pueden conectarse al módulo de muestreo y gestión de datos. También permite conocer el tiempo que el módulo no va a poder grabar muestras debido a la ejecución de cualquiera de los test citados.

- **Medición del tiempo de realización de los test:** Cada tipo de test lleva asociado un pin de la FPGA. Siempre que se esté ejecutando uno de los test, el pin asociado tomará el valor de '1', en cualquier otro caso tomará el valor de '0'.

- **Medición de los tiempos de ejecución de un refresco (sólo DDR):** Se van a realizar dos tipos de medidas para dimensionar correctamente la duración de los comandos de refresco. La primera medida consiste en la medición del tiempo que transcurre desde que el controlador solicita un comando de refresco hasta que éste finaliza. La segunda medida consiste en la medición del tiempo que transcurre desde que el comando de refresco empieza a ejecutarse hasta que éste finaliza. Cada medida lleva asociada un pin de la FPGA, que tomará el valor de '1' durante el transcurso del tiempo citado.

- **Medición del tiempo de acceso durante la escritura:** Se generará una señal PWM que aportará información sobre el porcentaje de tiempo en el que se accede a memoria durante la escritura. El periodo de la señal PWM se corresponde con 100 microsegundos en este caso. Es importante que el período sea lo suficientemente grande para poder aportar una información lo más representativa posible de la escritura (la ejecución de varios refrescos, la inclusión de varios microsegundos de muestreo, etc.). Durante el periodo seleccionado, se va a calcular el número de ciclos en los que no se escriben muestras en el interfaz estando esté en posición de recibirlos, esto es, no se deben contabilizar refrescos, ciclos de inicialización y finalización de los comandos de escritura, etc. Tras contabilizar los ciclos en los que no se escriben datos en el interfaz durante 100 us, se genera la onda PWM. Tomará el valor de '1' durante un número de ciclos de reloj igual a los ciclos contabilizados anteriormente, el resto de ciclos de reloj de los que consta el período de la señal, de 100 microsegundos, la onda PWM tomará el valor de '0'.

Las simulaciones que se han hecho a continuación se han realizado sobre el kit de Xilinx Spartan-3A/3AN FPGA Starter Kit (a no ser que se especifique lo contrario). Este kit consta de una FPGA Spartan3-700AN y de una memoria DDR2, la frecuencia del reloj empleado es de 125 MHz.

Simulación de las mediciones:

Simulaciones realizadas con 4 tarjetas I/O y 2 registros de estado.

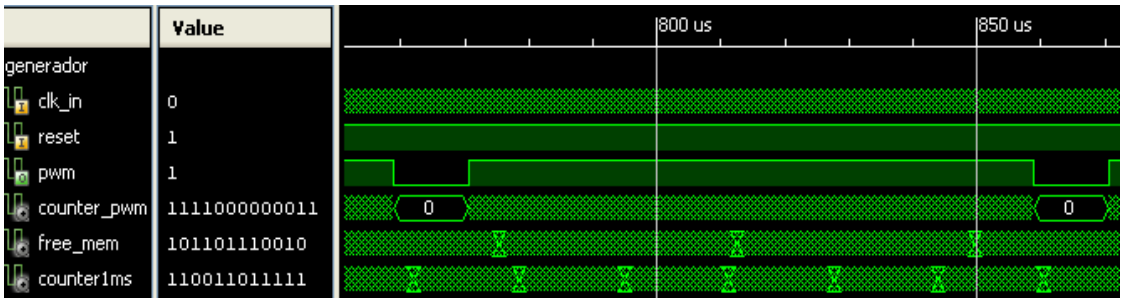

Ilustración 72: Simulación medición de acceso a la memoria durante la escritura

Se observa cómo PWM tiene efectivamente un período de 100 microsegundos. Además presenta un ciclo de trabajo del 88% (88 microsegundos). Si se tiene en cuenta que se están procesando 10 muestras por microsegundo, se puede deducir que sólo consumen un 100-88 = 12% de la capacidad real del interfaz (reloj de 125 MHz).

A continuación se mide el tiempo de realización de los test:

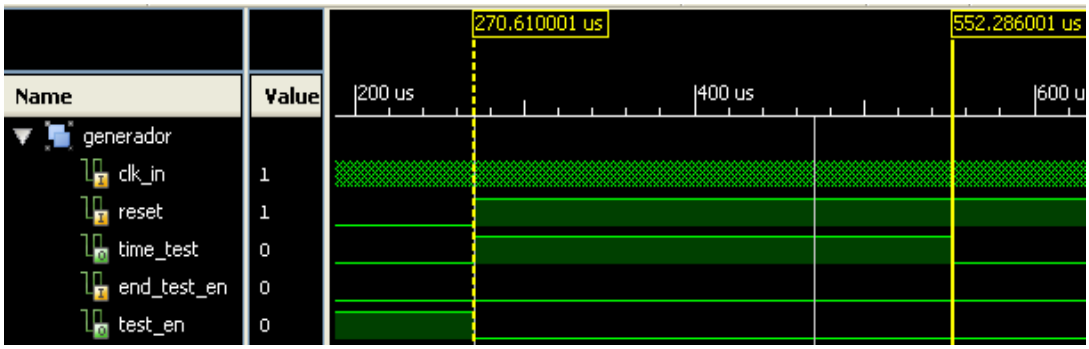

Ilustración 73: Simulación tiempo de realización de test_1

La medida obtenida en el test_1 es de 552,28 us – 270,61 us = 281,68 microsegundos.

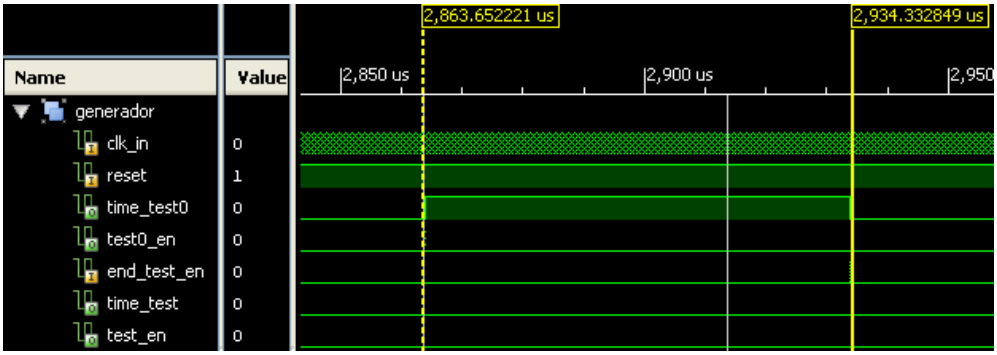

Ilustración 74: Simulación tiempo de realización de test_0

La medida obtenida en el test_0 es de 2934,33 us – 2863,65 us = 70,68 microsegundos.

Simulación de los comandos de refresco (Tarjeta Spartan3 – 1600 E):

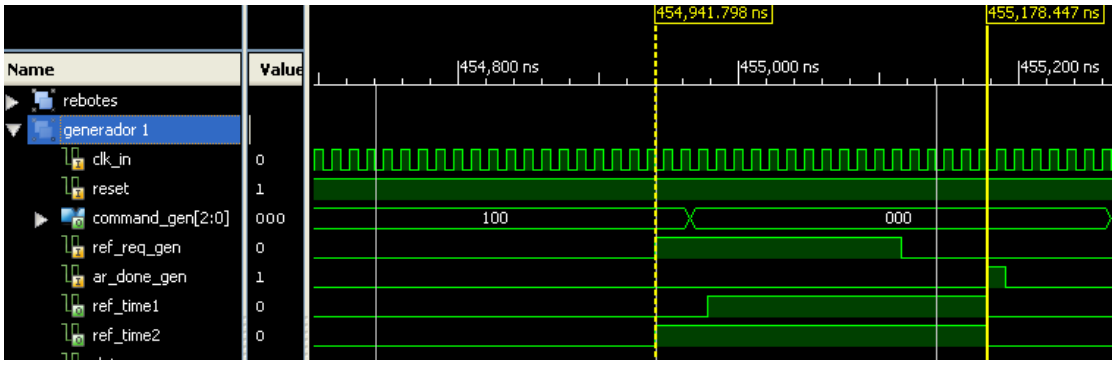

Ilustración 75: Simulación comando de refresco en DDR

Ref_time1 se corresponde con la segunda medida del comando de refresco. En este caso la medida es de: 455.178 ns – 454.978 ns = 200 nanosegundos. Tiempo estimado de duración de un comando de refresco.

Ref_time2 se corresponde con la primera medida del comando de refresco. En este caso la medida obtenida es de 455.178 ns – 454.941 ns = 237 nanosegundos. Tiempo estimado entre la petición de un refresco por el controlador y la finalización de éste.

Mediciones físicas de los tiempos de acceso.

Las mediciones físicas se realizarán con un osciloscopio. La idea es utilizar pines disponibles en la placa para facilitar las mediciones. Para ello habrá que hacer uso de los pines de la FPGA que estén conectados con los pines externos de la placa. Mediciones para 4 tarjetas I/O y 2 registros de estado.

A continuación se exponen las medidas realizadas con el osciloscopio:

Memoria DDR2 (125 MHz):

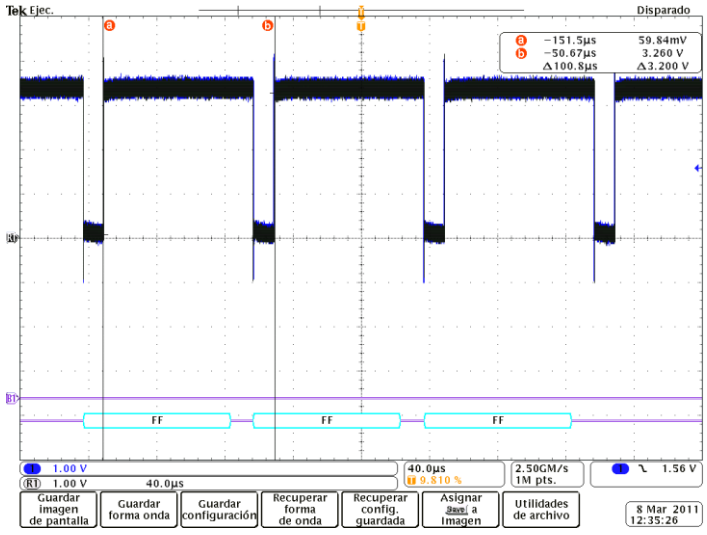

Ilustración 76: Medida acceso a memoria durante la escritura. DDR2

Tras posicionar el switch'0' en posición de escritura, el ciclo de trabajo de la onda PWM es del 89% (89 microsegundos). Por la tanto se deduce una capacidad de procesamiento de muestras en la escritura n, donde n = $(100 \times 10) \div (100 - 89)$ = 90.9 datos/microsegundo. Con reloj de 125 MHz.

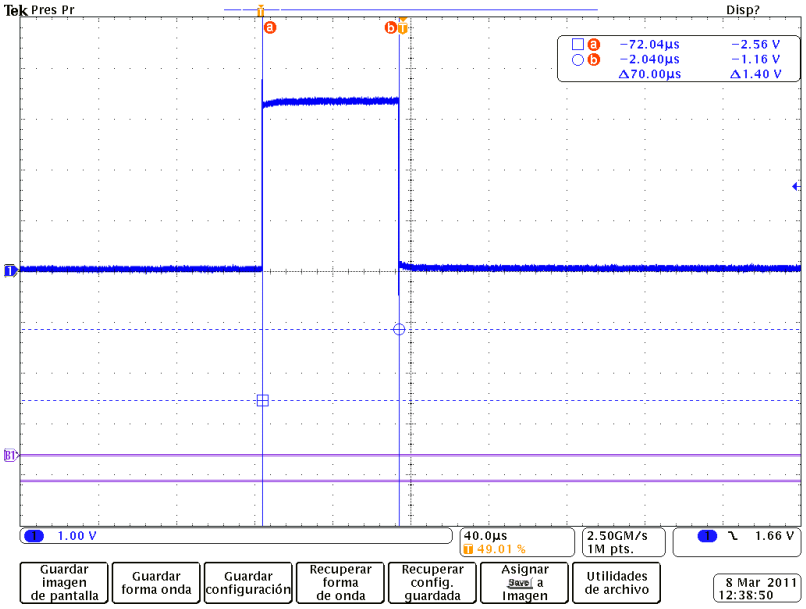

Ilustración 77: Medida inicialización de la memoria DDR2

La inicialización de la memoria (test_0) dura 70 microsegundos.

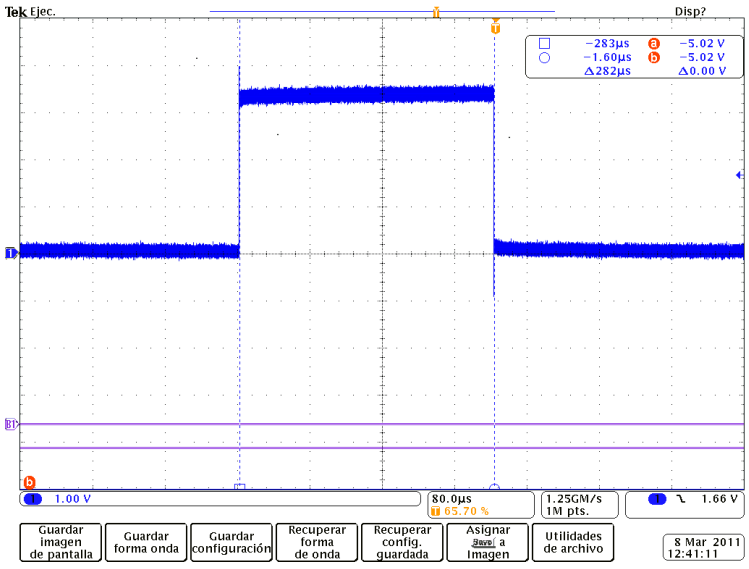

Ilustración 78: Medida test completo de la DDR2

El test completo de la DDR2 dura 282 microsegundos.

Memoria DDR (80 MHz):

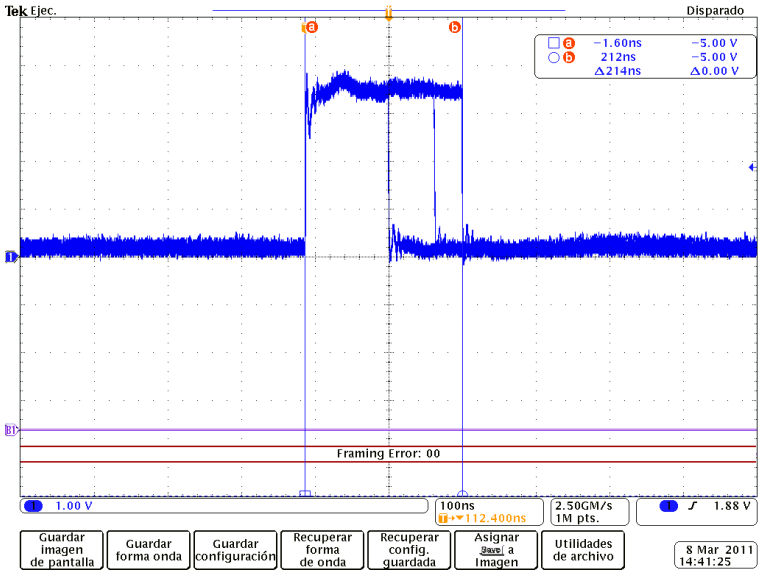

Ilustración 79: Duración de un comando de refresco en la DDR

Un comando de refresco dura aproximadamente 214 ns en el peor caso posible.

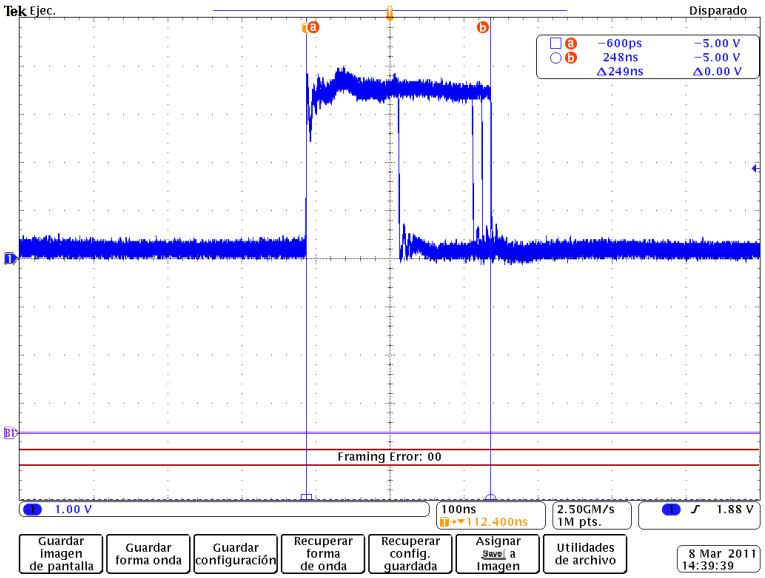

Ilustración 80: Tiempo entre la solicitud de un refresco y su finalización. DDR

Se tardan aproximadamente 249 ns (en el peor caso posible) desde que el controlador solicita un comando de refresco hasta que éste finaliza.

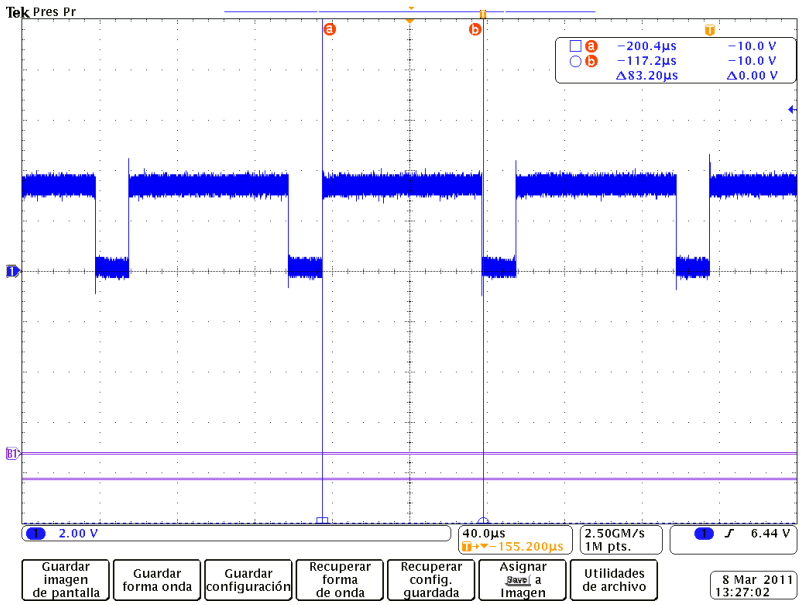

Ilustración 81: Medida de acceso a memoria durante la escritura. DDR

La señal presenta un ciclo de trabajo del 83.2% (83.2 microsegundos). Por la tanto, se deduce una capacidad de procesamiento de muestras en la escritura n, donde n = $(100 \times 10) \div (100 - 83.2)$ = 59.52 datos/microsegundo (desarrollo completo en la simulación). Con un reloj de 80 MHz.

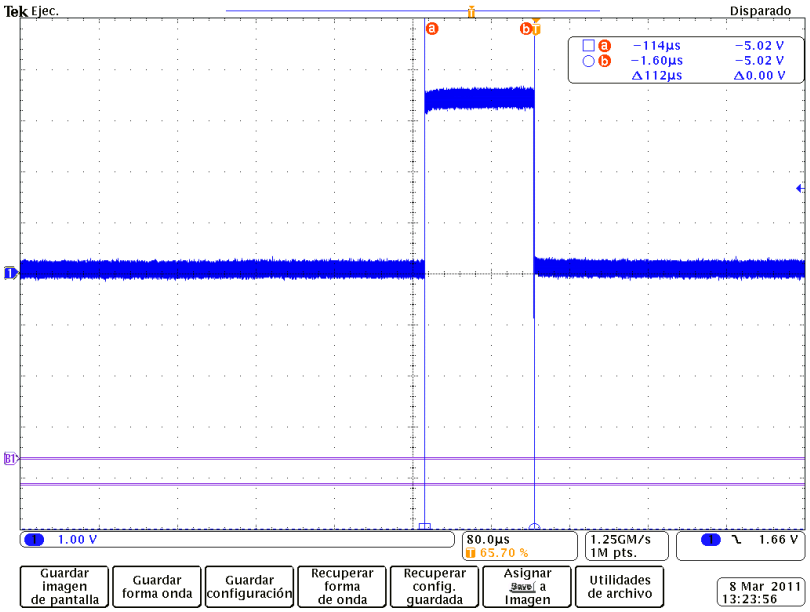

Ilustración 82: Medida inicialización de la memoria DDR

La duración de la inicialización es de 112 microsegundos.

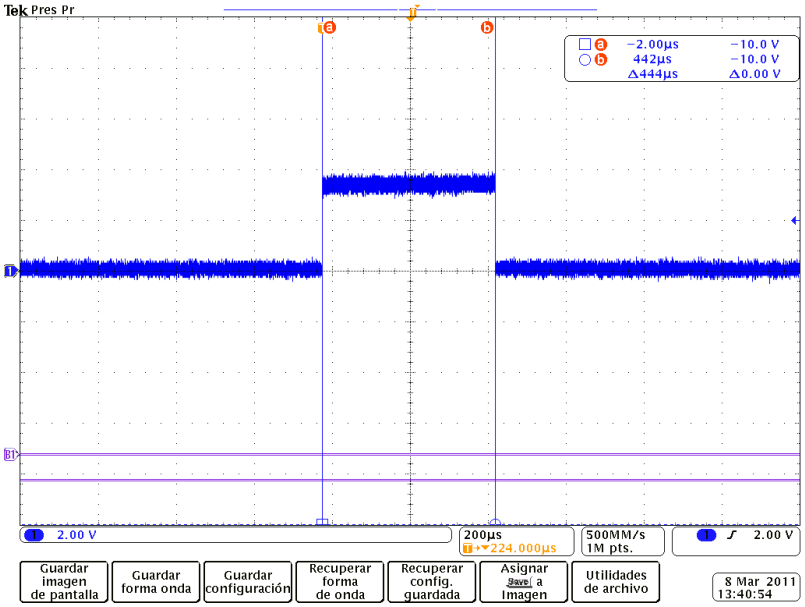

Ilustración 83: Medida test completo DDR

La duración del test completo es de 444 microsegundos.

Correspondencia entre la simulación y las medidas físicas:

Se puede deducir que el test_1 y el test_0 no presentan variaciones respecto a la simulación. Los comandos de refresco presentan una pequeña variación, debido a que la medida física hace referencia al peor caso posible (el tiempo de simulación apenas es de 3 milisegundos). Del mismo modo, la velocidad de escritura es simular a la obtenida en la simulación, cumpliéndose con el requisito de diseño: 10 muestras / microsegundo.

2.5 Resultados de síntesis.

Se van a proporcionar los resultados de síntesis del módulo "muestreo y gestión de datos en memoria" que emplea la memoria DDR (el que finalmente se implementará).

La FPGA presente en el diseño es la Spartan3–1600E. Dicha FPGA se monta sobre el "Spartan-3E FPGA Industrial Micromodule" de Trenz. Se va a sintetizar el diseño para 4 tarjetas I/O y 2 registros de estado.

Área:

Los resultados de área se obtienen del MAP con el software ISE.

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	1,244	29,504	4%	
Number of 4 input LUTs	1,369	29,504	4%	
Number of occupied Slices	1,585	14,752	10%	
Number of Slices containing only related logic	1,585	1,585	100%	
Number of Slices containing unrelated logic	0	1,585	0%	
Total Number of 4 input LUTs	1,405	29,504	4%	
Number used as logic	1,242			
Number used as a route-thru	36			
Number used for Dual Port RAMs	64			
Number used as Shift registers	63			
Number of bonded IOBs	251	304	82%	
IOB Flip Flops	17			
IOB Master Pads	1			
IOB Slave Pads	1			
Number of ODDR2s used	19			
Number of BUFGMUXs	2	24	8%	
Number of RPM macros	1			
Average Fanout of Non-Clock Nets	3.09			

Tabla 19: Área módulo de muestreo y gestión de datos en memoria

Máxima frecuencia:

Los resultados de frecuencia se obtienen del Place&Route con el software ISE. Es la máxima frecuencia para la que se garantiza un funcionamiento correcto del circuito. La frecuencia de funcionamiento en servicio va a ser de 80 MHz.

```
Design statistics:
   Minimum period:  12.282ns(1)    (Maximum frequency:  81.420MHz)
   Maximum path delay from/to any node:  11.406ns
   Maximum net delay:   1.638ns
```

Ilustración 84: Máxima frecuencia módulo de muestreo y gestión de datos en memoria

Como se puede observar, el diseño cumple con los requisitos de frecuencia.

3 MÓDULO DE LA DFT.

En este capítulo se va a desarrollar un módulo capaz de calcular una componente armónica de una señal, tomándose como referencia la transformada discreta de Fourier (DFT). A este algoritmo basado en la DFT se le denominará DFT específica (se centra en uno de los armónicos en vez de todo el espectro) o módulo de la DFT.

La DFT específica permite conocer el valor eficaz de cualquier componente armónica presente en una señal. La utilización del algoritmo en este caso, está motivada por dos aspectos fundamentales en el convertidor:

- **Detección de averías en los IGBTs del inversor**. La potencia de salida del inversor trifásico es constante, al producirse una avería en un IGBT se produce un desequilibrio en dicha potencia (deja de ser constante). La componente fundamental de la corriente de entrada al inversor permite conocer la existencia de averías en los IGBTs del inversor. Cuando el valor eficaz de la componente fundamental de la corriente supere un valor umbral, se interpretará como una avería de al menos uno de los IGBTs (IGBT abierto).

- **Cálculo de la Distorsión armónica total (THD)**: para realizar dicho cálculo es preciso conocer el valor eficaz de la componente fundamental. La DFT específica permite obtener el valor eficaz de la componente fundamental.

El módulo de la DFT debe poder calcular el algoritmo para un número de entradas configurables. El valor del módulo se va a proporcionar en un bus de datos común a todos los canales. El esquema se muestra en la Ilustración 85:

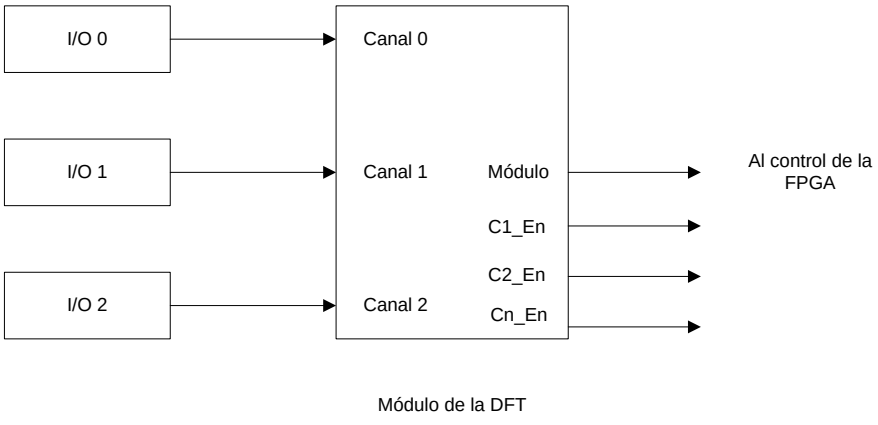

Ilustración 85: Esquema general de la DFT específica

Para dimensionar el módulo de la DFT, se ha hecho uso del popular software MATLAB:

- Se ha diseñado un modelo de simulación con la herramienta SIMULINK, con el que se ha verificado el funcionamiento del algoritmo.

- Para el dimensionado del algoritmo se ha utilizado la herramienta Fixed Point de MATLAB.

3.1 Requisitos de diseño.

El principal requisito, es realizar un diseño modular en base a las siguientes características fundamentales:

- Se desea extraer el valor de una componente de frecuencia dada.
- El número de señales sobre las que se calcula la DFT específica debe ser configurable. Del mismo modo, el armónico a calcular para cada señal debe ser igualmente configurable: fundamental, doble de la fundamental, etc. (se debe poder calcular más de un armónico para la misma señal si es preciso). Con al menos, una capacidad mínima de 5 señales y/o armónicos.
- El sistema debe ser configurable para trabajar con dos tipos de señales mutuamente excluyentes: bien señales cuyo armónico fundamental sea de 50 Hz (sistema Europeo) o bien señales cuyo armónico fundamental sea de 60 Hz (sistema de EE.UU.).
- Las señales a las que se aplica el algoritmo de la DFT específica no van a tomar valores imaginarios, únicamente presentan parte real. Las señales de entrada se proporcionan en un formato de 13 bits con signo. En caso que la señal sobre la que se aplica el algoritmo no presente signo, el bit 13 permanecerá siempre a '0'.
- La frecuencia de muestreo se fija en 125 KHz.
- La medida de la DFT específica (número complejo) debe proporcionarse siempre en notación polar. Además, el argumento es despreciable, únicamente se requiere el módulo. Esto es, debe proporcionarse éste último o bien el valor eficaz, o bien el valor eficaz al cuadrado.
- Optimización del consumo de recursos Hardware. Se debe tener en cuenta que la finalidad de éste módulo no es otra que la de integrarse en la plataforma de control del convertidor. La FPGA sobre la que se implementa el módulo DFT presenta pocos recursos disponibles y requiere una optimización del consumo hardware.

3.2 Estudio del algoritmo con MATLAB.

El software de matemáticas MATLAB se ha empleado inicialmente, para realizar una primera aproximación a la DFT, es decir, comprender el funcionamiento del algoritmo. Se ha diseñado un modelo mediante la herramienta SIMULINK para verificar la comprensión del algoritmo.

Posteriormente, se ha hecho uso de la herramienta Fixed Point. La funcionalidad de la herramienta es digitalizar cualquier modelo de SIMULINK, es decir, obtener un modelo basado en números binarios. La utilidad de la herramienta es clara, dimensionar el hardware que finalmente se implementará en la FPGA:

- Número y tamaño de acumuladores.
- Número de multiplicadores dedicados de la FPGA.
- Número y ancho de posiciones de memoria necesarias. Este parámetro determina el número de Block RAMs necesarias de la FPGA.
- Selección del tamaño de la ventana, etc. (en los siguientes apartados se explicarán los parámetros que definen la DFT).
- Una mala selección del tamaño del acumulador puede redundar en un mal funcionamiento de la DFT a pesar de que el algoritmo se esté realizando de una forma aparentemente correcta. Igualmente, la selección de estos parámetros influye directamente en la resolución de la DFT.
- El último estudio realizado con MATLAB ha sido la adaptación de la medida. El algoritmo de la DFT proporciona como salida un número complejo. El módulo de dicho número es proporcional al valor eficaz del armónico analizado. Por tanto es necesaria una lógica que sea capaz de calcular módulo del número complejo proporcionado por la DFT.

3.2.1 Descripción de los principales parámetros de la DFT

A continuación se van a analizar los principales parámetros de la DFT con el objeto de entender el funcionamiento del algoritmo. Una vez se comprenda el funcionamiento del algoritmo se podrá crear un modelo fiel con la herramienta SIMULINK.

Se recuerda que la expresión general por la que se rige la DFT es la siguiente:

$$X(k) = \sum_{n=0}^{N-1} x(n)\ W_N^{kn} \qquad 0 \le k \le N-1$$

Donde:

$X(k)$ es la secuencia de N números complejos obtenida de calcular la DFT de la secuencia $x(n)$ de longitud N.

El factor de fase $W_N = e^{\frac{-j\,2\,\pi}{N}}$, es el factor de rotación en el plano complejo. Si se toma N = 4, los posibles valores de W_N^{kn} desde n= 0 hasta n = N - 1 se pueden observar en la imagen adjunta:

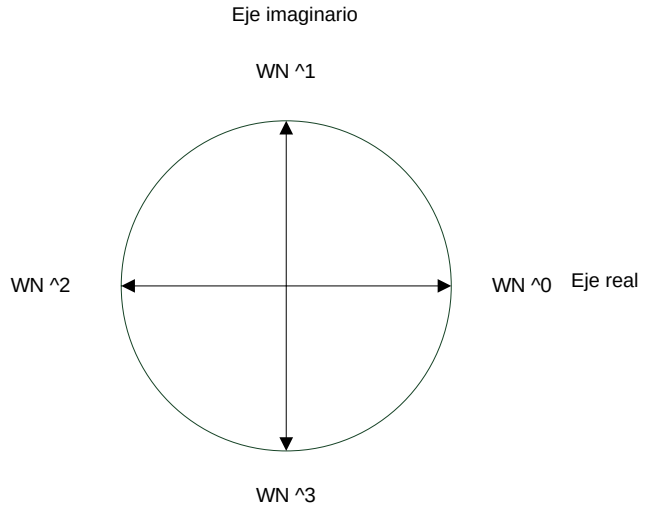

Ilustración 86: Factores de fase en el plano complejo

El algoritmo de la DFT, al igual que todos los algoritmos de tratamiento digital de señales, no trata la señal en tiempo continuo, sino en tiempo discreto. Es decir, se trabaja con las muestras de la señal a tratar. En este proyecto, la frecuencia de muestreo viene impuesta a 125 kHz (3.1). Esto supone una restricción importante en el diseño, ya que restringe los grados de libertad para la selección de los parámetros más importantes del algoritmo.

Los parámetros a los que se ha hecho referencia son los siguientes:

Frecuencia de muestreo (f_{mstr}): viene impuesta a 125 kHz.

Ancho de la ventana (A_W): es la ventana de frecuencia con la que se calcula la DFT. Se debe tomar como máximo la mitad de la frecuencia de la señal a medir para evitar problemas de "aliasing" según el Teorema de Nyquist. Teniendo en cuenta que la velocidad de procesamiento no es un factor crítico en el diseño, se ha escogido una ventana de 10 Hz.

Número de puntos, N: es el número de puntos de la DFT, esto es, el número de muestras con el que trabaja el algoritmo. N afecta sobre todo a la resolución de la salida de la DFT.

La fórmula que relaciona el número de puntos con la frecuencia de muestreo y el ancho de la ventana es la siguiente:

$$N = {f_{mstr}}/{A_w} => N = {125\ KHz}/{10\ Hz} = 12500\ puntos$$

Para facilitar la identificación del resto de parámetros se va a hacer uso de la notación matricial para la DFT:

$$\begin{pmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N & W_N^2 & \dots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \dots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \dots & W_N^{(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{pmatrix}$$

Esta notación simplifica la identificación del funcionamiento del algoritmo. Cada posición del vector $X(k)$ representa un armónico de la señal a la que se aplica la DFT. Para saber en qué posición del vector $X(k)$ están los armónicos a calcular se aplica la siguiente ecuación:

$$K = \frac{(f_{armónico}\ N)}{f_{mstr}} => sustituyendo\ queda: k = \frac{(f_{armónico}\ 12500)}{125\ KHz}$$

$$k = \frac{f_{armónico}}{10}$$

Donde $f_{armónico}$ es la frecuencia del armónico a calcular.

De aquí se deduce que las posiciones de $X(k)$ se corresponden con:

$X(0)$ = valor de la componente continua.

$X(1)$ = valor del armónico coincidente con la frecuencia de la ventana seleccionada.

$X(k)$ = valor del armónico coincidente con k veces la frecuencia de la ventana seleccionada.

Para una señal cuyo armónico fundamental es de 50 Hz y usando los parámetros calculados anteriormente se tiene:

X(0) = valor de la componente continua de la señal.

X(1) = es el valor del armónico de 10 Hz (nulo en éste caso).

***X(5)* = es el valor de la componente fundamental (50 Hz).**

***X(10)*= es el valor de la componente del doble de la fundamental (100 Hz).**

Tras esta primera aproximación, queda claro que sólo es necesario calcular los subíndices del vector *X(k)* que aporten información sobre los armónicos que se desea calcular, reduciéndose de este modo, el número de operaciones que inicialmente se necesitan para el cálculo del algoritmo. Es decir, el presente módulo se centra en el cálculo de uno de los subíndices del vector *X(k)* (un solo armónico).

3.2.2 Modelado de la DFT específica con la herramienta SIMULINK.

Introducción a la herramienta:

SIMULINK es un entorno interactivo para modelar una amplia variedad de sistemas dinámicos, pudiendo ser estos lineales, no lineales, discretos, de tiempo continuo y sistemas mixtos. Permite realizar las operaciones mediante diagramas de bloques. Adicionalmente dispone de una herramienta de simulación con la que visualizar los resultados.

Es también un sistema abierto, que permite al usuario escoger, adaptar y crear componentes o subsistemas. SIMULINK se apoya en el software MATLAB.

Por tanto, SIMULINK y MATLAB proveen un entorno integrado para construir modelos versátiles y simular modelos dinámicos, diseñando y testeando ideas nuevas.

A modo de ejemplo, se van a exponer los cálculos realizados para las señales cuyo armónico fundamental sea de 50 Hz.

Identificación de parámetros:

Si tomamos uno de los subíndices del vector X(k), por ejemplo el a, se obtiene :

$$X(a) = \begin{pmatrix} 1 & W_N^a & W_N^{2a} & \dots & W_N^{a\,(N-1)} \end{pmatrix} \begin{pmatrix} x(0) \\ x(1) \\ \vdots \\ \vdots \\ x(N-1) \end{pmatrix}$$

Éstas son las operaciones necesarias para calcular cualquier armónico.

En realidad, el factor de fase W_N no es más que un número complejo expresado en notación exponencial e^{jw}, por tanto, mediante la fórmula de Euler se puede representar un número complejo en notación exponencial como:

$$e^{jw} = \cos(w) + j\,sen(w)$$

De esta manera se puede descomponer el factor de rotación como la suma de un seno y un coseno. Esta descomposición es clave para comprender como se ha diseñado el algoritmo. Sustituyendo en W_N queda:

$$W_N = e^{\frac{-j\,2\,\pi}{N}} = \cos\left(\frac{2\,\pi}{N}\right) - j\,sen(\frac{2\,\pi}{N})$$

Si se tiene en cuenta que la secuencia x(n) de longitud N, son las muestras de entrada a la DFT, se puede deducir, que las muestras están distribuidas en el tiempo en función de la frecuencia de muestreo (125 kHz). Por tanto el cálculo del algoritmo va a ser secuencial, es decir, se va a procesar cada punto de la DFT específica a razón de un periodo de muestreo de 8 microsegundos.

Se va a dividir el cálculo en dos partes, por un lado la parte real y por otro, la parte imaginaria. Cada parte va a contar con un acumulador, encargado de acumular el de cada punto de la secuencia, hasta que se complete la secuencia de N puntos. Momento en el que se iniciará el cálculo de una nueva secuencia $x(n)$ de N puntos. El valor que va a tomar cada acumulador se obtiene de la siguiente fórmula:

$$Acum_{re} = \sum_{n=0}^{n=N-1} \cos\left(\frac{2\pi\,n\,a}{N}\right) * \; x(n)$$

$$Acum_{im} = \sum_{n=0}^{n=N-1} -j\sin\left(\frac{2\pi\,n\,a}{N}\right) * \; x(n)$$

Primer esquema para el cálculo de la DFT para un solo armónico:

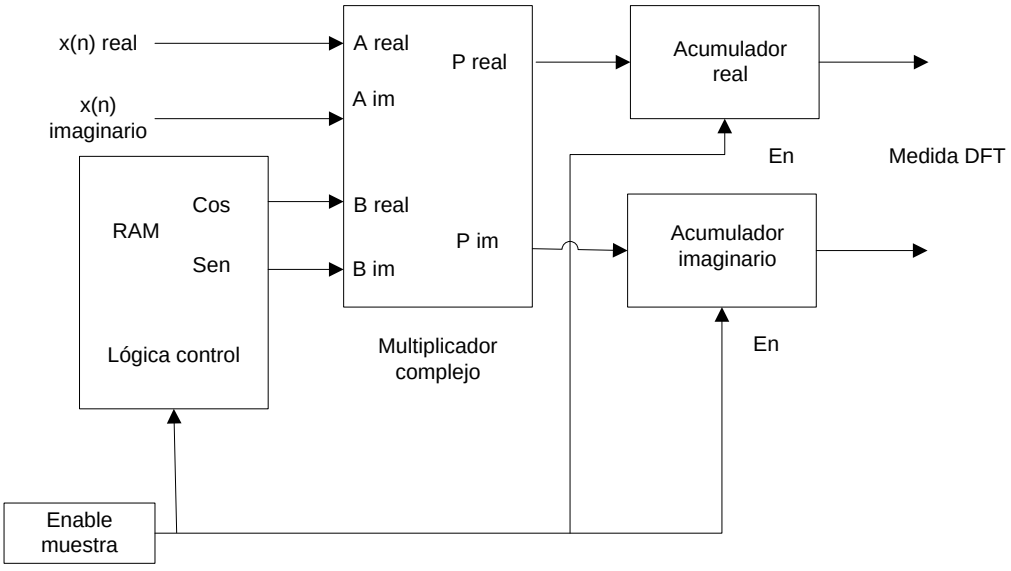

Ilustración 87: Diagrama de bloques primera aproximación DFT específica

Tras obtener el primer esquema se van a realizar las simplificaciones con las que finalmente se generará el modelo en SIMULINK:

Las muestras de entrada $x(n)$, únicamente van a tomar valores reales, es decir, la parte imaginaria va a ser siempre 0. Se va a necesitar un multiplicador para multiplicar las muestras de entrada por los factores de rotación. En el modelo se han empleado dos multiplicadores (uno para cada parte del factor de rotación).

Para la lógica de control de los factores de rotación (seno y coseno), se han tabulado y se ha creado un contador para generar el valor de entrada a una función que genera el seno y el coseno. Este contador se incrementa con una frecuencia igual a la de muestreo y se incrementa de *a* en *a*, donde a es el subíndice del vector $X(k)$.

Para comprender el funcionamiento de la lógica de control, se retoma la expresión de W_N calculada anteriormente:

$$W_N = e^{\frac{-j\,2\,\pi}{N}} = \cos\left(\frac{2\,\pi}{N}\right) - j\,sen(\frac{2\,\pi}{N})$$

Si se tienen en cuenta todos los posibles valores de W_N para el cálculo del subíndice a del vector $X(k)$ se tiene:

$$W_N^{h\,a}\ \text{, desde } h = 0 \text{ hasta } h = (N-1)$$

El significado de la expresión anterior, no es otro que el factor de rotación recorre los valores del seno y el coseno entre $0\ y\ 2\pi$ incrementándose cada nuevo punto. El parámetro *a* marca la frecuencia con la que se da una vuelta completa al periodo de las dos funciones trigonométricas. Teniendo en cuenta los parámetros calculados hasta ahora, el factor de rotación va a equivaler a un seno y un coseno con una frecuencia $(10\ a)\ Hz$.

A modo de ejemplo, se demostró anteriormente que en $X(5)$, estaba contenida la componente fundamental de 50 Hz. Por lo tanto, la frecuencia del factor de rotación es igual a: $10 * 5 = 50$ Hz.

De lo anterior se deduce que el algoritmo diseñado consiste en multiplicar la secuencia de entrada (muestras) por un factor de rotación complejo. Este factor de rotación se descompone en la suma de un seno imaginario y un coseno real, con una frecuencia igual a la frecuencia del armónico que se desea calcular. La suma compleja de todas las multiplicaciones de la secuencia de entrada y el factor de rotación, se corresponde con la medida de la DFT específica (número complejo). El módulo de este número complejo es proporcional a la amplitud de la frecuencia estudiada.

Creación del modelo:

Una vez que se han definido todos los parámetros anteriores, es posible definir un modelo con la herramienta SIMULINK que represente la DFT. La utilidad de éste modelo, radica en la facilidad con la que se pueden modificar los parámetros calculados anteriormente y observar como repercuten en el comportamiento del algoritmo. Es decir, el modelo de SIMULINK ayuda a dimensionar el algoritmo. Además, ofrece una idea aproximada del hardware que requiere el cómputo del

algoritmo y de la posible estrategia de control que mejor se adapte a las necesidades de diseño.

El modelo está estructurado en varias partes:

- Unidad básica de la DFT específica.
- Unidad de adaptación de la medida de la DFT específica a un valor físico.
- Unidad de control del coeficiente de rotación.

Como nota aclarativa, no se pretende diseñar un modelo complejo en exceso. Más bien uno fácil de entender y manipular. Donde se requiere diseñar un control riguroso y eficaz, es en el circuito implementado en la FPGA.

Unidad básica de la DFT:

Es, básicamente, el diagrama de bloques de la primera aproximación al que se ha sustituido el multiplicador complejo por dos multiplicadores convencionales. Se puede observar que el acumulador no tiene señal de "clear". El modelo sólo pretende ser útil para calcular una sola vez la DFT en cada simulación.

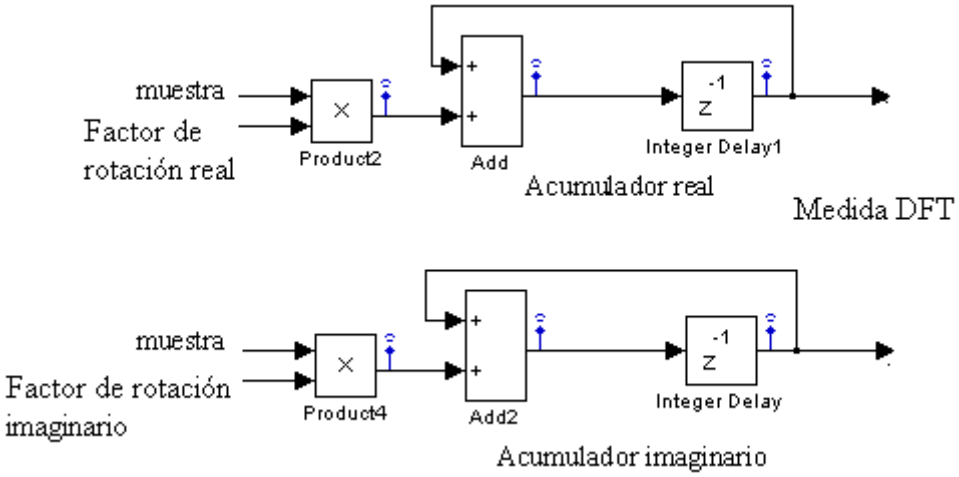

Ilustración 88: medida básica de la DFT específica. SIMULINK

Adaptación de la medida DFT a un valor físico:

Como se puede constatar en el esquema anterior, la DFT proporciona en su salida un número complejo en notación binomial con la forma:

$$Medida_{DFT} = accum_{re} + jaccum_{im}$$

Los requisitos de diseño piden el cálculo del módulo. Con lo cual queda:

$$|Medida_{DFT}| = \sqrt{accum_{re} + accum_{im}}$$

Tras estos cálculos, se obtiene la medida realizada por el algoritmo. El último paso es adaptar el valor de la medida de a la señal física que se está muestreando. Lo que se pretende es establecer una relación entre el valor eficaz del armónico y el módulo de la medida de la DFT específica. La siguiente ecuación proporciona la relación buscada:

$$V_{efArmónico} = \frac{|Medida_{DFT}|\sqrt{2}}{N}$$, o expresado en valor de pico:

$$V_{picoArmónico} = \frac{|Medida_{DFT}|\ 2}{N}$$, donde N es el número de puntos.

Éstos son los valores reales que realmente representan el valor eficaz y el valor de pico de los armónicos calculados. Sin más, se expone el esquema de SIMULINK que implementa la adaptación de la medida al valor físico de la señal:

Por coeficiente de adaptación se entiende $\frac{\sqrt{2}}{N}$ en el caso de que se calcule el valor eficaz del armónico o $\frac{2}{N}$ en caso de que se calcule el valor de pico.

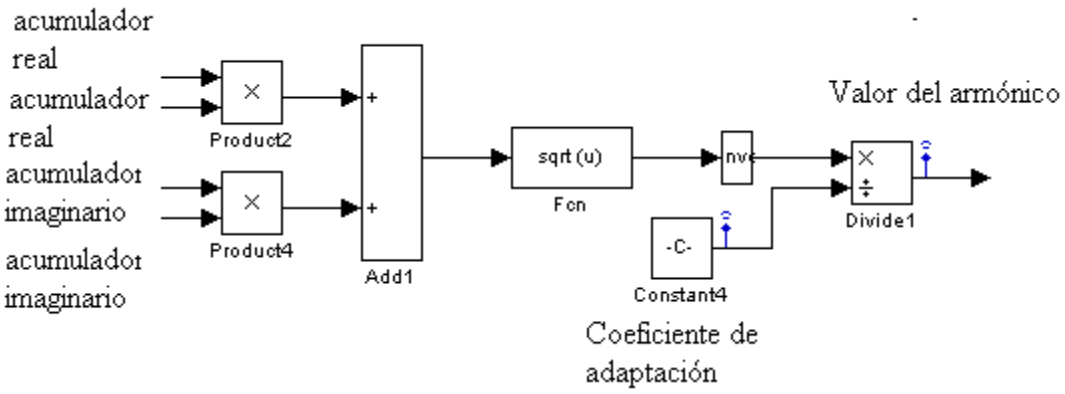

Ilustración 89: Adaptación física de la medida DFT específica. SIMULINK

Control del coeficiente de rotación:

Esquema de control:

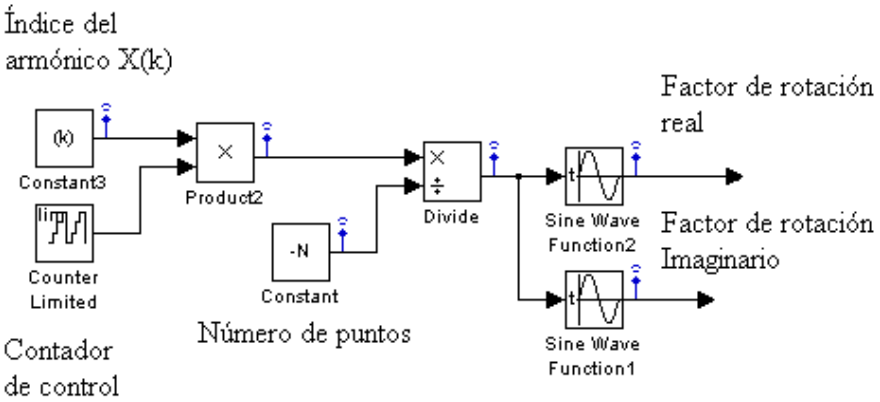

Ilustración 90: Control de los factores de rotación. SIMULINK

El funcionamiento es sencillo. Se genera un coseno y un seno, que se corresponden con los valores, real e imaginario respectivamente, del factor de rotación. El índice del vector $X(k)$, que como se ha demostrado anteriormente, se corresponde con el armónico a calcular, marca la frecuencia del factor de rotación. En resumen, se genera:

$$W_N^{h\,a} \text{ , desde } h = 0 \text{ hasta } h = (N-1)$$

El incremento del contador se corresponde con la frecuencia de muestreo. El límite del contador se corresponde con el número de puntos, N.

Simulación del modelo:

Se pretende simular el modelo de la DFT específica para comprobar su funcionamiento. Para realizar las medidas necesarias, se ha añadido al modelo bloques de SIMULINK con los que poder visualizar y cuantificar señales (scopes y displays). Como señal a medir, se ha utilizado una onda cuadrada (50 Hz) con la siguiente descomposición armónica:

La amplitud A en el ejemplo, toma el valor de 1000 Vpico.

$$f(x) = A\,\frac{4}{\pi}\left(\frac{sen(x)}{1} + \frac{sen(3x)}{3} + \frac{sen(5x)}{5} + \cdots\right)$$

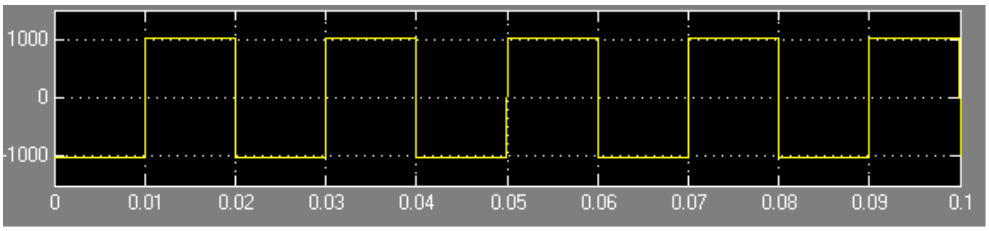

Ilustración 91: Onda cuadrada para la simulación. SIMULINK

Medida del primer armónico:

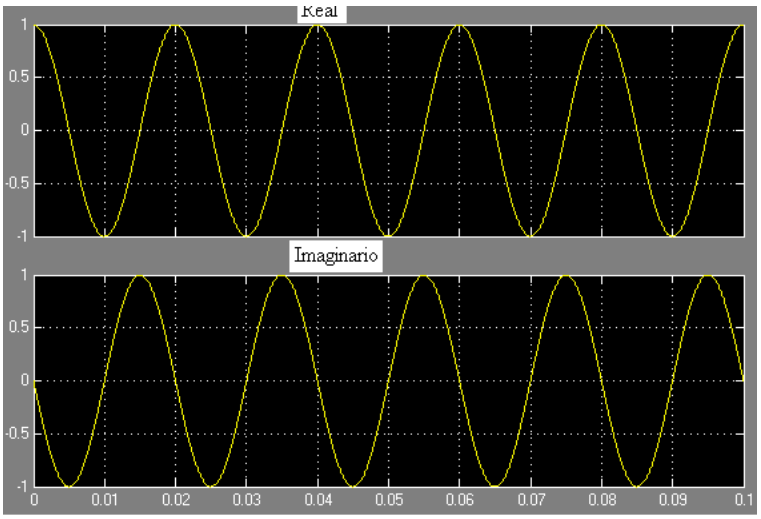

Ilustración 92: Factores de rotación para armónico de 50 Hz. SIMULINK

La medida del valor de pico proporcionada por la DFT es de 1273. Si se realiza el cálculo teórico, el valor esperado es de:

$$f_{fundamental} = 1000\frac{4}{\pi} = 1273{,}23\ Vpico$$

Medida del segundo armónico:

Como se observa en el desarrollo armónico de la onda cuadrada, no existen armónicos pares: el valor de pico proporcionado por la DFT es de 0,48 (aproximadamente 0).

Medida del tercer armónico:

Al medir el tercer armónico de la señal, la frecuencia de los factores de rotación toma el valor de 150 Hz (coincidente con la frecuencia del tercer armónico). El valor de pico aportado por la DFT, es de 424,4. El valor teórico esperado es de:

$$f_{3armónico} = 1000\frac{4}{\pi}\frac{1}{3} = 424{,}41\ Vpico$$

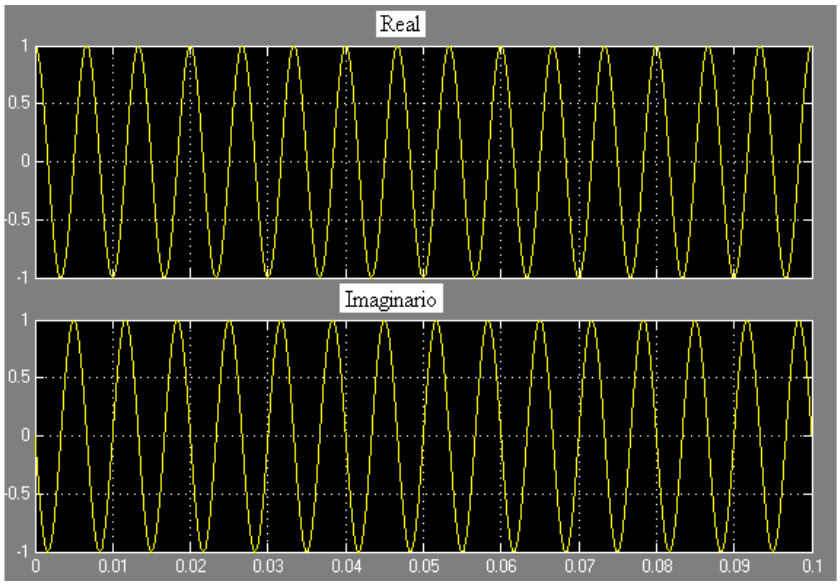

Ilustración 93: Factores de rotación para armónico de 150 Hz. SIMULINK

A modo de ejemplo se muestran los acumuladores real e imaginario para el tercer armónico:

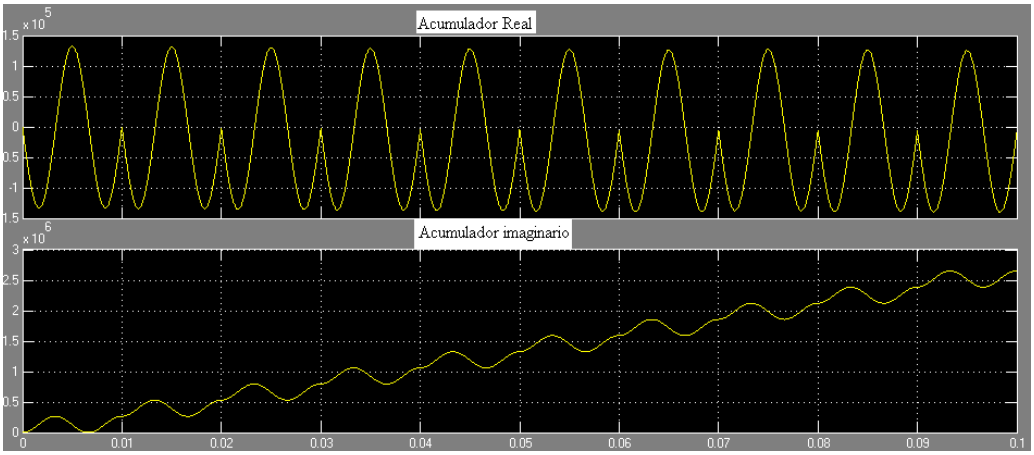

Ilustración 94: Medida de la DFT específica para armónico de 150 Hz. SIMULINK.

El modelo al completo se adjunta en el CD proporcionado. Es posible simular el modelo y consultar las formas de onda medidas con los "scopes".

3.2.3 Dimensionamiento del modelo mediante la herramienta Fixed Point:

Introducción a la herramienta:

SIMULINK Fixed Point permite usar las capacidades intrínsecas de cálculo en coma fija de MATLAB en el diseño de sistemas de control y tratamiento de señal con funciones aritméticas. Con el propósito de crear flujos de trabajo más eficientes, permite transformar diseños en coma flotante de SIMULINK, Stateflow, etc, a diseños en coma fija.

La herramienta permite trabajar con señales y variables de 1 a 128 bit, lo que permite acelerar los modos de simulación. Permite desarrollar e implementar algoritmos para DSPs y µPs, etc.

En este proyecto se ha hecho un uso limitado de las múltiples opciones/aplicaciones de la herramienta. Por lo que se considera que no es necesario profundizar en las posibilidades que brinda la herramienta más allá de lo estrictamente necesario.

La funcionalidad que se ha cubierto con Fixed Point es el dimensionamiento de los recursos que finalmente se implementarán en la FPGA: acumuladores, multiplicadores, memoria, etc.

Digitalización del modelo:

Una de las múltiples funcionalidades de Fixed Point es la conversión de un modelo de SIMULINK a un modelo totalmente digitalizado, esto es, un modelo basado en señales y variables binarias. Éste es el paso previo al dimensionamiento del algoritmo.

Las diferentes etapas con las que se digitaliza el modelo se muestran a continuación:

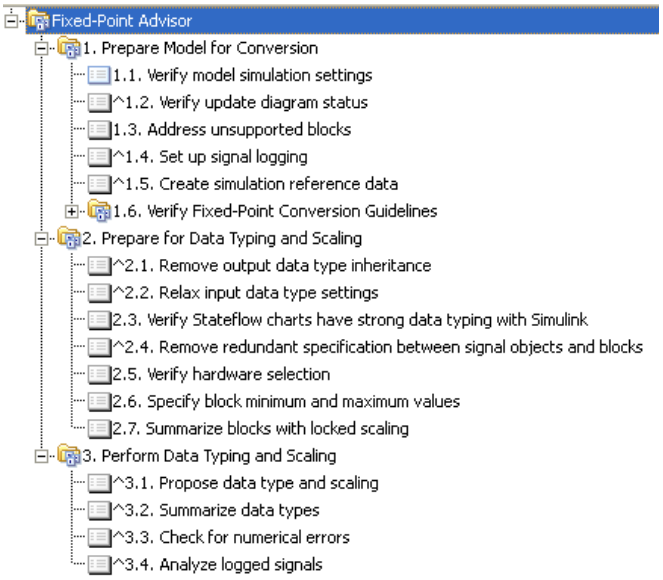

Ilustración 95: pasos para la digitalización de un modelo SIMULINK.

Se debe tener en cuenta que la implementación no va a realizarse con MATLAB, sino que se va a utilizar el SOFTWARE ISE de Xilinx. En realidad se han seguido todos los pasos mostrados en la Ilustración 95, pero los más interesantes de cara al lector son el "3.1 Propose data type and scaling" y el "3.3 Check for numerical errors".

En el apartado 3.1, la herramienta solicita al usuario que defina tipos binarios para las señales presentes en el modelo. El formato utilizado ha sido el fixed point (A, B, C) donde:

A indica el signo de la señal, '1' señal con signo, '0' señal sin signo.

B indica el número de bits que definen la señal (incluye A y C).

C representa el número de bits para la coma fija.

En el caso de que el usuario defina un tipo con una resolución inadecuada, la herramienta propone tipos alternativos que se adapten a la funcionalidad del modelo.

Recommended Data Type Summary		
Block	**Recommended data type**	Rationale
dft_sim/Sine Wave_DTC_Out1	fixdt(1,13,0)	Maintained current settings; overflow is possible.
dft_sim/Sine Wave1_DTC_Out1	fixdt(1,13,0)	Maintained current settings; overflow is possible.
dft_sim/Sine Wave2_DTC_Out1	fixdt(1,13,0)	Maintained current settings; overflow is possible.
dft_sim/Signal Generator_DTC_Out1	fixdt(1,13,2)	Improved precision; overflow is not possible based on min/max range.
dft_sim/Sum of Elements : Output	fixdt(1,13,2)	Improved precision; overflow is not possible based on min/max range.
dft_sim/Product5	fixdt(1,32,13)	Improved precision; overflow is not possible based on min/max range.
dft_sim/Divide	fixdt(1,32,28)	Maintained current settings; overflow is not possible based on min/max range.
dft_sim/Sine Wave Function2_DTC_Out1	fixdt(1,14,12)	Maintained current settings; overflow is not possible based on min/max range.

Tabla 20: Extracto de los valores propuestos por la herramienta para la digitalización del modelo

En el apartado 3.2, la herramienta cuantifica el error de medida producido por la digitalización el circuito. El usuario puede seleccionar las señales que desee medir y la herramienta simulará el modelo previo a la digitalización y el modelo digitalizado (ante unos estímulos definidos por el usuario). La medición del error se realiza de

forma gráfica. Ante errores inadmisibles, es posible rediseñar el modelo con valores fixed point alternativos y volver a medir los resultados. A continuación se expone un aviso simulado en el que la herramienta ha descubierto un error de saturación en un divisor (el divisor utilizado en la generación de los factores de rotación):

Blocks with numerical errors		
Block	**Error Occurrence**	**Analyze Signals**
dft_sim/Divide	Overflow:5833 Saturation:0 ParamSaturation:0 DivisionByZero:0	unnamed ('Inport 1'): View Results... unnamed ('Inport 2'): View Results... unnamed ('Outport 1'): View Results...

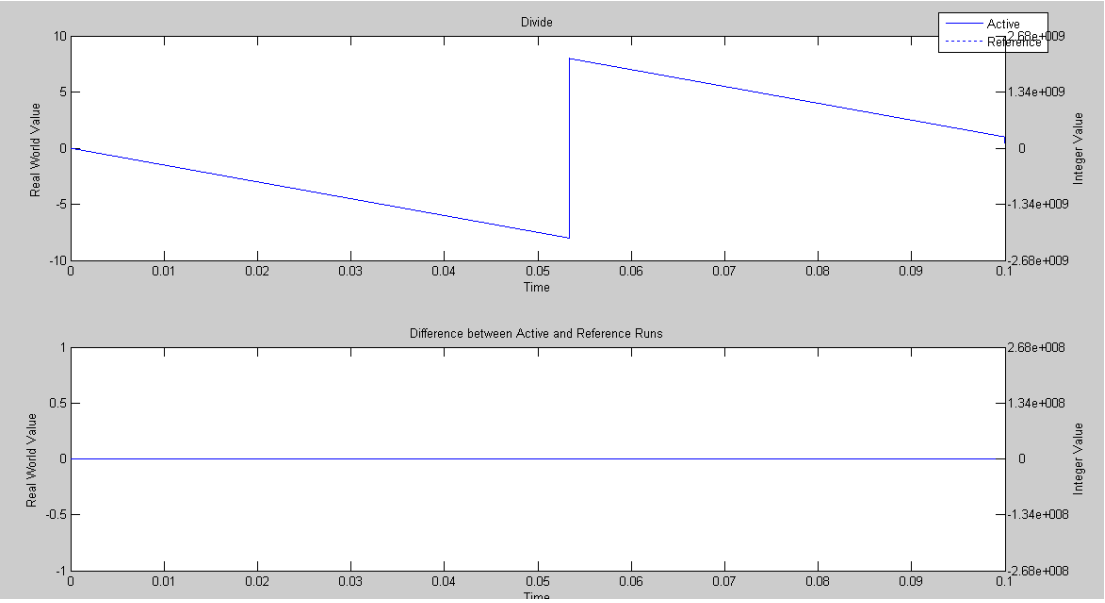

Ilustración 96: Saturación de un divisor durante la digitalización

El error en este caso, se produce al dividir por 0.

Dimensionamiento del hardware:

Una vez que se ha digitalizado el modelo, es posible realizar un estudio sobre el coste en hardware que supone el cálculo de la DFT. Así como el dimensionamiento de la lógica necesaria para el cálculo del algoritmo.

La arquitectura actual de las FPGAs, está especialmente diseñada para facilitar ciertas operaciones básicas en el procesado digital de la señal. Una de las operaciones más habituales de todas, es la acumulación de la multiplicación por coeficientes. La arquitectura de la FPGA presenta un multiplicador implementado a la salida de cada Block RAM. Si en la Block RAM se almacenan los coeficientes de la multiplicación, puede resultar especialmente atractivo el uso de la citada arquitectura.

En este caso en concreto, lo que se va a almacenar en la Block RAM son los valores del factor de rotación (seno y coseno). Se pretende implementar algo con un principio de funcionamiento similar al mostrado en el esquema adjunto:

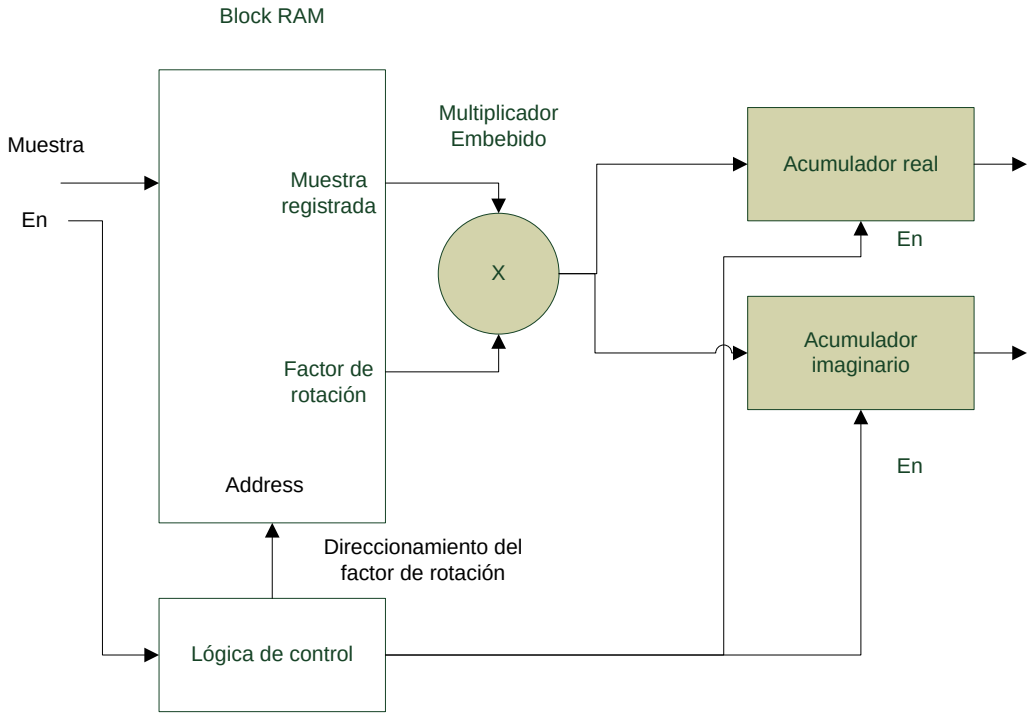

Ilustración 97: Arquitectura de la FPGA para multiplicación con acumulación

El objetivo del dimensionado es cumplir, en la medida de lo posible, el requisito de optimización del hardware consumido. Para lograr el objetivo propuesto, se ha decidido utilizar como máximo uno de los multiplicadores implementados de la FPGA y una Block RAM.

En realidad, existen múltiples opciones a la hora de trabajar con senos y cosenos a parte de la utilización de tablas en memoria, como el desarrollo en series de Taylor, etc. Pero en relación precisión/área están, claramente, en desventaja.

Primer estudio. Características del diseño:

- Las muestras de entrada presentan un formato de 13 bits con signo.
- Ventana de 10 Hz (A_W).
- El número de puntos de la DFT es de 12500 (12500 acumulaciones).

Primer estudio. Dimensionado:

El primer paso es dimensionar los factores de rotación. El factor de rotación complejo se divide en un seno imaginario y un coseno real. Los valores del seno y el coseno están comprendidos entre 1 y -1. Además, se sabe que el armónico de menor frecuencia que se requiere calcular es de 50 Hz que como se demostró anteriormente, es coincidente con la frecuencia más lenta del factor de rotación. Precisamente, el peor caso posible para el dimensionamiento del seno o el coseno, es la frecuencia más lenta que se requiere representar, ya que obliga a poder representar una mayor cantidad de

puntos diferentes (pertenecientes al mismo periodo) a lo largo de la misma ventana fija de 10 Hz.

Si en una ventana $A_W = 10\ Hz$ se van a realizar 12500 rotaciones del vector, significa que el seno y el coseno van a realizar un periodo completo (de 0 a 2π) cada:

$\cos(wh) = cos\left(\frac{2\pi}{N} k\ h\right)$, donde k para el peor caso posible (50 Hz) = 5. Con lo que queda $\cos(wh) = \cos\left(\frac{2\pi}{12500}\ 5\ h\right) = \cos\left(\frac{2\pi}{2500} h\right)$, desde $h = 0$ hasta $(12500 - 1)$.

Es decir, cada 2500 puntos de la DFT, el coseno habrá realizado un periodo completo (razonamiento análogo para el seno). Por tanto, se van a necesitar n bits para representar los factores de rotación donde n se obtiene de:

$2^n = 2500$, despejando n queda, $n = \log_2 2500 = 12$

Además, en los 12 bits está incluido el bit de signo y se precisa de un bit adicional que actúe como parte entera. De este modo se consigue representar el valor -1 y el +1, con lo que queda n = 13. Para dar un mayor margen a la resolución del seno y el coseno, se ha decidido emplear un bit adicional, haciendo un total de 14 bits.

Si se toma el resultado en función del hardware disponible en la FPGA, se deduce lo siguiente. Se van a necesitar 5000 posiciones de memoria RAM para almacenar los valores del seno y el coseno (2500 posiciones en RAM cada uno). Cada posición requiere un mínimo de 14 bits para lograr la resolución calculada.

Las Block RAMs disponibles en la FPGA Spartan3-1600E se pueden configurar para diferentes anchos de palabra. Los anchos de palabra son potencias de 2 y varían desde 1 a 32 bits. El ancho de palabra que mejor se ajusta al diseño (14 bits) es el de 16 bits. Además, la Block RAM presenta un tamaño total de 16384bits, lo que da un total de 1Kword de 16 bits por palabra.

Los factores de rotación precisan de un total de $n = \frac{5000}{1024}$ = 4.88 = 5 Block RAMs. No es necesario seguir dimensionando el algoritmo, 5 Block RAMs es un coste elevado en memoria. Se debe encontrar una solución alternativa que permita reducir el número de posiciones en memoria requeridas por los factores de rotación.

Optimización de los factores de rotación:

Se han contemplado dos posibles opciones para reducir el consumo de memoria.

Simetría de las funciones trigonométricas. Se utilizan las siguientes relaciones trigonométricas (en radianes):

$sen(x) = \cos\left(x - \frac{\pi}{2}\right)$, reduciéndose en 2500 el número de posiciones en memoria.

Simetría del seno en cuarto de periodo: reduciéndose en $\frac{3*2500}{4}$ las posiciones en memoria. Lo que da un total de $5000 - 2500 - 1875 = 625$ posiciones.

Otra opción que se ha contemplado es la reducción del número de puntos.

Reducción del número de puntos (N).

La reducción del número de puntos conlleva una serie de ventajas significativas: reducción de la memoria, reducción de los acumuladores (real, imaginario, etc), sumadores, etc.

Por el contrario, al reducir el número de puntos calculados, se corre el riesgo de perder resolución. Para minimizar la pérdida de resolución, se ha decidido reducir el número de puntos N a un número divisor de 12500. El máximo número de puntos que cumple con la condición de ser divisor de N y requerir un número de posiciones en memoria menor o igual a 1024 es:

$$\frac{5000\ posiciones}{a} \leq 1024\ posiciones$$, lo que da un a = 5.

$$\frac{12500\ puntos}{5} = 2500$$ puntos (N).

De este modo se precisan un total de 1000 posiciones en memoria (el cálculo es análogo al descrito en: Primer estudio. Dimensionado). La utilidad de que el nuevo número de puntos sea divisor de 12500, brinda la posibilidad de realizar la media de *a* muestras (el factor *a* calculado anteriormente) que precede al cómputo de cada punto, tal como se muestra en el siguiente diagrama de flujo:

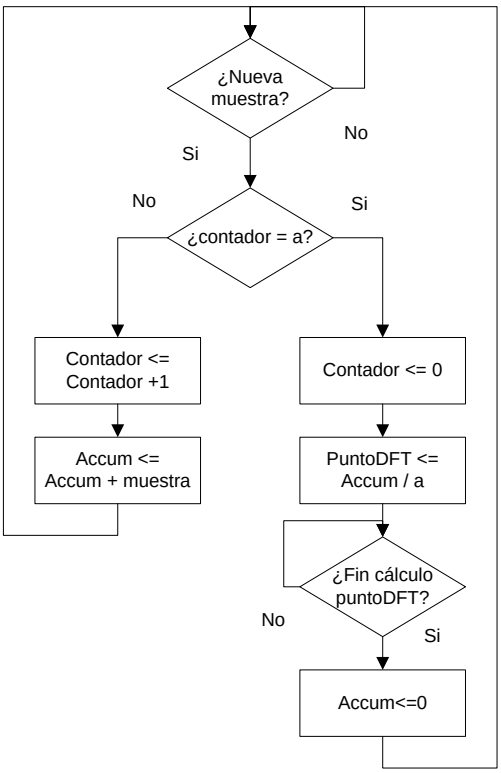

Ilustración 98: Diagrama de flujo reducción del número de puntos en la DFT específica

Se concluye que para reducir los factores de rotación se va a optar por disminuir el número de puntos de la DFT específica de 12500 a 2500. Cada punto proporcionado a la DFT es la media de 5 muestras. De este modo, no se desprecia ninguna muestra, minimizándose la reducción de la precisión.

Segundo estudio características del diseño:

- La entrada presenta un formato de 13 bits con signo. Es el resultado de la media de 5 muestras.
- Ventana de 10 Hz (A_W).
- El número de puntos de la DFT es de 2500 (2500 acumulaciones).

Segundo estudio. Dimensionado:

La parte que calcula la media de 5 muestras no está incluida en el modelo. El dimensionado se describirá en el apartado Implementación de la DFT.

El número de posiciones de memoria necesarias es de 1000, 500 para el seno y 500 para el coseno. Los factores de rotación presentan un ancho de 11 bits, 9 para la coma fija (500 valores precisan 9 bits), 1 para el signo y 1 como parte entera para representar el 1 y el -1. El consumo es de una Block RAM.

Multiplicador: presenta el formato a*b = p. Donde: a es la media de 5 muestras (13 bits con signo), b es el factor de rotación (11 bits con signo). Por tanto, p se representa con 22 bits: 1 de signo, 12 enteros y 9 para la coma fija. Se utilizan 22 y no 23 bits en el multiplicador debido a que el valor máximo que van a tomar los factores de rotación va a ser +1 o -1. La parte decimal no se puede despreciar, debido a que al acumular 2500 multiplicaciones puede dar lugar a que la parte decimal genere algún valor entero. Si se tiene en cuenta que la salida de los multiplicadores embebidos de la FPGA presenta un ancho de 36 bits, se dispone de un margen holgado.

Acumuladores real e imaginario: van a realizar 2500 acumulaciones de la multiplicación. El peor caso, desde el punto de vista de la saturación, es el cómputo de la DFT de una señal cuadrada con una amplitud máxima que esté en fase con uno de los factores de rotación. La amplitud máxima de una señal de 13 bits con signo es de $2^{13-1} = 4095$ Vpico.

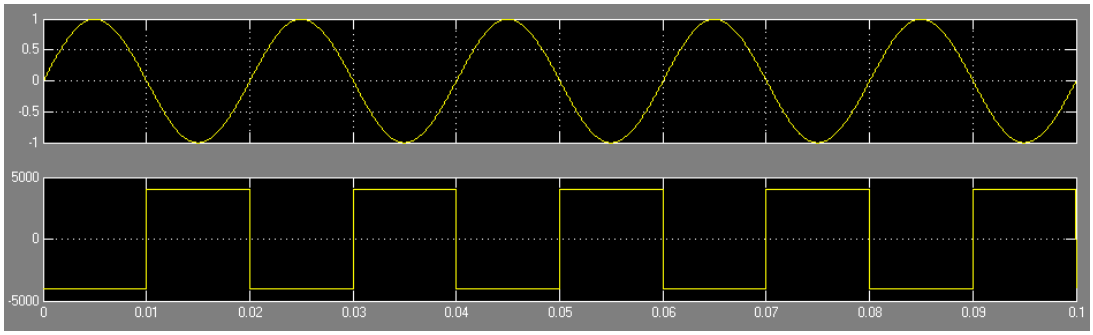

Ilustración 99: Onda cuadrada en fase con la parte imaginaria del factor de rotación

El acumulador presenta un ancho de 33 bits, 9 decimales, 1 de signo y 23 enteros.

Módulo: cuyo formato es $|M| = \sqrt{Re^2 + Im^2}$, se ha decidido dimensionar en el capítulo "Implementación de la DFT". Los valores que se proporcionan en el modelo son meramente orientativos, pero de ninguna manera, son asumibles para su implementación en la FPGA. Esto es debido, principalmente, a los tamaños sugeridos para el hardware (sumadores cercanos a los 50 bits, multiplicadores que superan con mucho, los 36 bits, etc.).

Conclusiones del dimensionado:

Se han hecho dos estudios más en función del número de puntos N de la DFT (consultar ficheros). Reducir el número de puntos reduce el tamaño de todos los circuitos aritméticos (acumuladores, sumadores, multiplicadores, etc). A continuación se muestra una tabla donde se realiza una comparativa entre el primer estudio, el segundo estudio y un estudio adicional (tercer estudio). Para el tercer estudio se toman 2500 puntos y una resolución de 16 bits para los factores de rotación. El objetivo es comparar la variación del error máximo (absoluto y relativo) en los acumuladores real e imaginario en función del número de puntos y de la resolución para los factores de rotación:

Característica	Primer estudio	Segundo estudio	Tercer estudio
Número de puntos	12500	2500	2500
Muestras de entrada	fi(1, 13, 0)*	fi(1, 13, 0)*	fi(1, 13, 0)*
Frec. de muestreo	8 microsegundos	40 microsegundos	40 microsegundos
Factores de rotación	fi(1, 14, 12)*	fi(1, 11, 9)*	fi(1, 16, 14)*
Ancho del multiplicador	fi(1, 25, 12)*	fi(1, 22, 9)*	fi(1, 27, 14)*
Ancho de los acumuladores	fi(1, 40, 12)*	fi(1, 33, 9)*	fi(1, 38, 14)*
Máximo error absoluto en el acumulador**	24902 unidades	1450 unidades	4998 unidades
Máximo error relativo en el acumulador***	0,00038%	0,00834%	0,00001%

Tabla 21: Comparativa para el dimensionado. Fixed Point

*El término fi(a, b, c) hace referencia al tipo de dato binario asociado al elemento de la columna de la izquierda, donde:

- a indica el singo, 1 con singo, 0 sin signo.
- b indica el ancho total del dato, incluido el bit de signo (en caso de haberlo).

- c hace referencia al número de bits para representar la coma fija.

**El máximo error absoluto se toma como la máxima diferencia entre el acumulador teórico y el acumulador digitalizado.

***El máximo error relativo se toma del siguiente coeficiente:

$$ErrorRel_{max}(\%) = \frac{\left|MedidaDFT_{teórica} - MedidaDFT_{digital}\right|}{MedidaDFT_{teórica}} * 100$$

El estudio se ha hecho suponiendo el peor caso posible para uno de los acumuladores. La señal sobre la que se calcula la DFT específica es una onda cuadrada de amplitud máxima (4095 unidades para 13 bits con signo) y en fase con uno de los acumuladores. El armónico a calcular es el de 50 Hz (cuya amplitud teórica sería de 5214 unidades) en los 3 estudios.

En la Ilustración 100 se muestra la ventana de trabajo en la que la herramienta Fixed Point indica los errores (gráfica de abajo) durante la digitalización del modelo. La ilustración muestra el acumulador real del segundo estudio, cuando el factor de rotación real está desfasado 90º respecto de la onda cuadrada sobre la que se aplica la DFT específica.

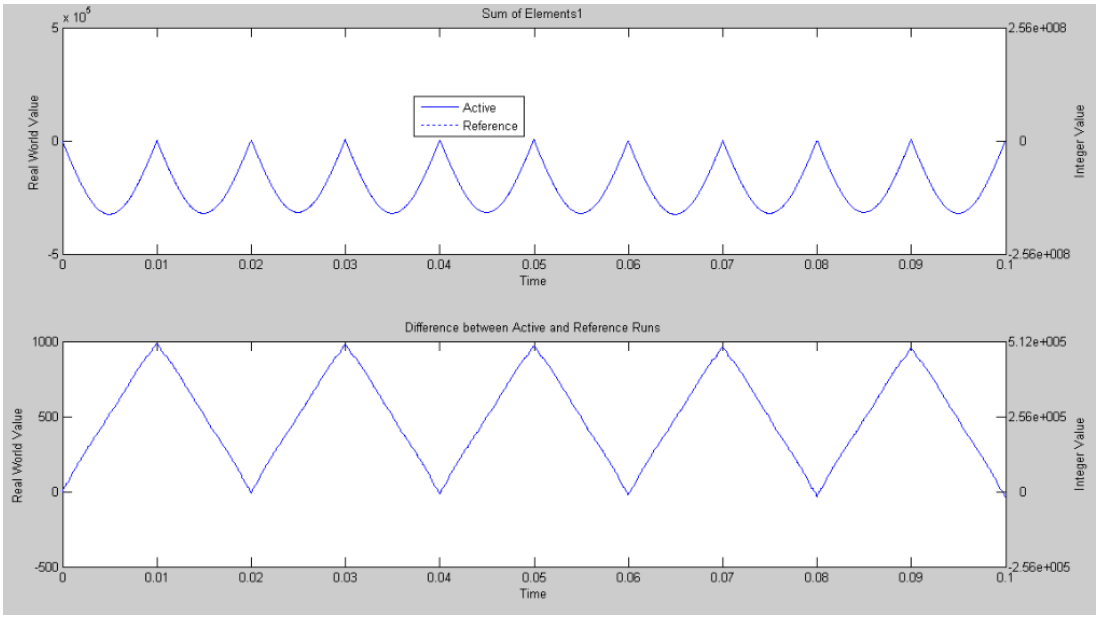

Ilustración 100: Medición de errores durante la digitalización mediante la herramienta Fixed Point

Los resultados obtenidos en el segundo estudio, se adaptan perfectamente a los requisitos planteados al inicio de diseño, sin perder una resolución significativa. Por lo que se ha decidido tomar el segundo estudio como referencia para la implementación de de la DFT en la FPGA.

3.3 Implementación de la DFT

En este apartado se va a diseñar un circuito que implemente el algoritmo de la DFT específica sobre la FPGA. Se va a tomar como referencia para el diseño, el modelo de SIMULINK Fixed Point con ciertas mejoras y/o añadidos:

- Se va a optimizar el cálculo del módulo de la DFT específica a partir de los acumuladores real e imaginario, incluyéndose la adaptación de la medida a la señal física muestreada (valor eficaz al cuadrado).
- Se va a incluir la arquitectura que calcula la media de n (5) muestras, con objeto de reducir el número de puntos (de 12500 a 2500).
- Diseño de un control que permita gestionar el cálculo simultáneo del algoritmo para un número programable de señales y/o armónicos (con al menos una capacidad mínima de 5).
- Se debe optimizar el consumo hardware. Como objetivo, se ha propuesto el uso de un único multiplicador implementado y una única Block RAM (común a todos los bloques de DFTs específicos).

Diagrama de bloques del módulo de la DFT para dos señales:

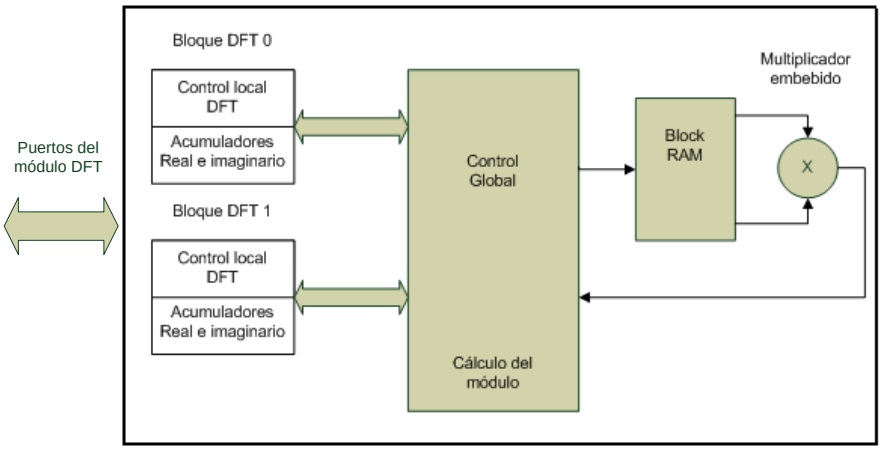

Ilustración 101: Diagrama de bloques módulo de la DFT para el cálculo de dos DFTs específicas

Puertos del módulo de la DFT:

A continuación se muestran los puertos del módulo de la DFT (Ilustración 101).

Puerto	Tipo	Descripción
ClkIn	In	Reloj de entrada al módulo.
ResetIn	In	Reset global para el módulo. Activo por nivel alto.
SampleDftEn	In	Bus de habilitaciones (enables). Indica que el canal

Puerto	Tipo	Descripción
[NumDFT - 1 downto 0]		de BusDftIn direccionado por SampleDftEn, está listo para su lectura. (cada 8 microsegundos aprox.)
BusDftIn [NumDFT - 1 downto 0]	In	Buses de muestras (13 bits cada muestra). Cada canal del bus se corresponde con un bloque local de DFTs.
EndDftEn [NumDFT - 1 downto 0]	Out	Bus de habilitaciones (enables) de finalización. Indica que DftModule está listo para su lectura. Nuevamente, cada canal del bus se corresponde a un bloque local.
DftModule [Accum_W - 1 downto 0]	Out	Medida realizada por DFT module. El bus es común para todos los bloques de locales de DFTs.

Tabla 22: Puertos del módulo de la DFT

NumDFT indica el número de señales y/o armónicos para los que se está calculando el algoritmo.

3.3.1 Bloques locales para el cómputo de la DFT:

El módulo de la DFT está dividido en dos bloques fundamentales, como se puede observar en la Ilustración 101:

- Bloque de control global, encargado de gestionar el hardware común a todos los bloques locales de DFTs (el multiplicador y la Block RAM).
- Bloque local para cada DFT, en el que están los acumuladores real e imaginario, el control del algoritmo, etc. Este último bloque es el que se va a tratar en este apartado.

Acumuladores real e imaginario:

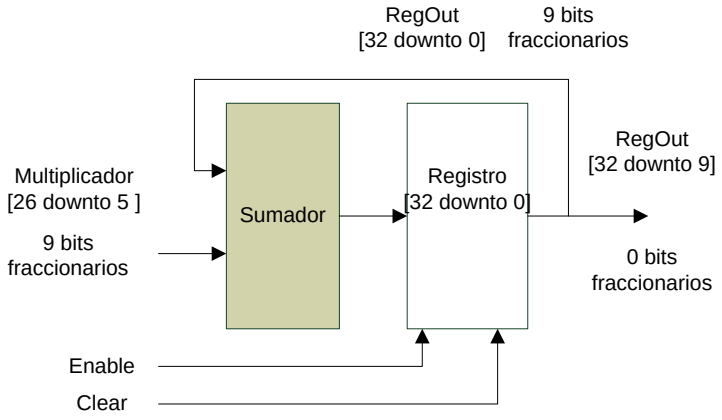

Ilustración 102: Implementación acumuladores real e imaginario

Los acumuladores se han diseñado en base a los resultados obtenidos en el estudio Fixed Point. Se acumulan 9 bits que representan la parte fraccionaria. El cálculo de cada punto se realiza en el control global. Una vez que finaliza el cálculo de un punto, se acumula en el correspondiente acumulador real o imaginario (enable). Una vez que

finaliza el algoritmo, solo se calcula el módulo en base a la parte entera. Al finalizar el cálculo del módulo en el control global, se inicializa a 0 el acumulador (clear).

Control local:

La lógica para implementar el control local de cada DFT específica se describe con el siguiente diagrama de flujo:

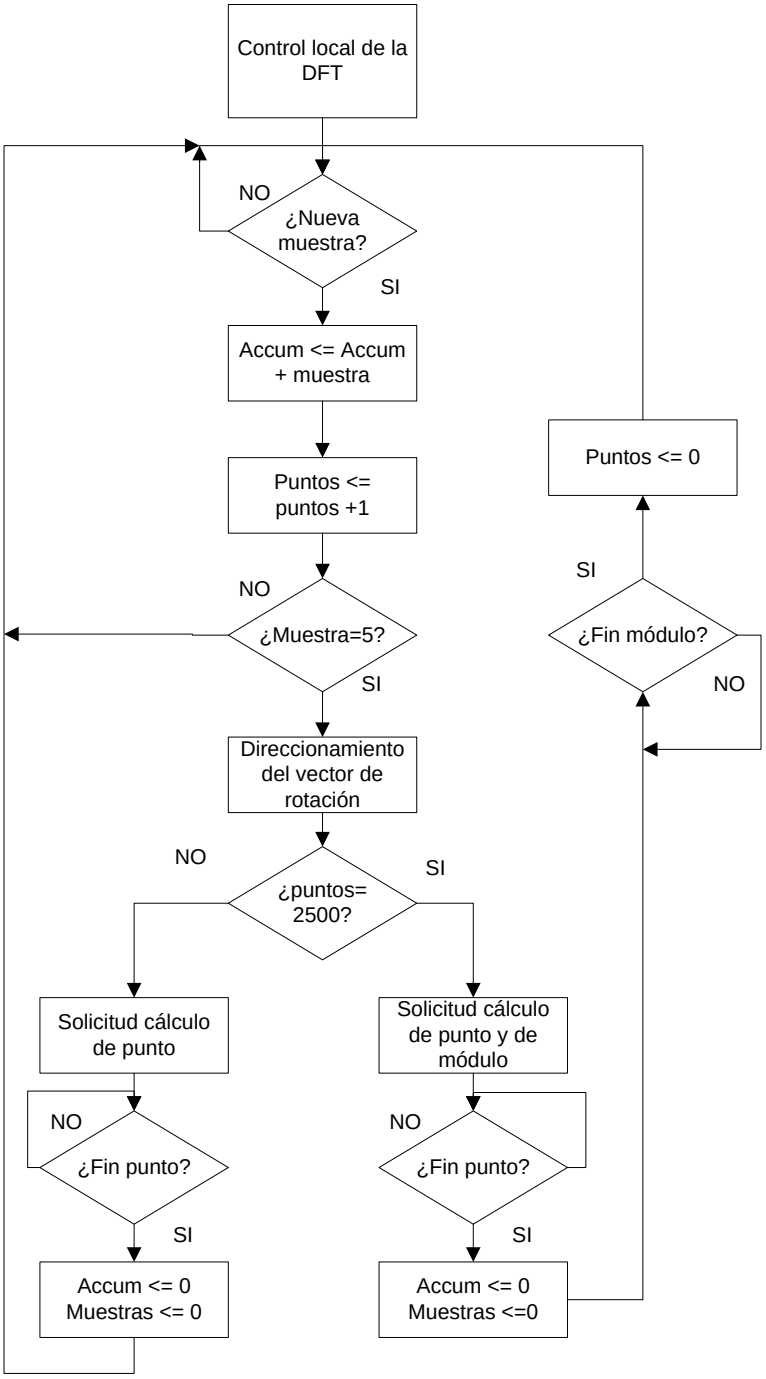

Ilustración 103: Diagrama de flujo control local de la DFT

En el control local se genera el direccionamiento para la Block RAM de los factores de rotación, se realiza la media de 5 muestras, se gestionan las peticiones de cálculo de

punto o de módulo al control global y se genera la lógica que gestiona el inicio y el fin del algoritmo (número de puntos procesados).

El direccionamiento de los factores de rotación se realiza con un contador. Dependiendo del armónico que se quiera calcular, el contador se incrementará en más o menos unidades a razón de: $Incre_{Con} = \frac{armónico}{Fundamental}$

El bus de direcciones de la Block RAM para los factores de rotación, se toma directamente de la salida del contador.

Al haber variado el número de puntos de la DFT de 12500 a 2500, también podría haber cambiado la relación entre los armónicos y su posición en el vector $X(k)$. A continuación se calcula la relación:

$K = \frac{(f_{armónico}\ N)}{f_{mstr}}$, para la fundamental de 50 Hz: $k = \frac{50*2500}{25k} = 5$

Se ve que la relación entre los armónicos y $X(k)$ no ha cambiado en absoluto. Un resultado lógico, ya que en la misma proporción que ha disminuido el número de puntos (5), ha disminuido la frecuencia de muestreo (5).

3.3.2 Control global de la DFT

Esta entidad gestiona el hardware común a todos los bloques de DFTs específicas, es decir, el control de la Block RAM y del multiplicador implementado.

El control global de la DFT específica está constituido por tres bloques: Un único acumulador empleado para el cálculo del módulo, un proceso para la gestión de las comunicaciones con los bloques locales de DFTs específicas y una máquina de estados para el gobierno del multiplicador y la Block RAM.

Arquitectura de la Block RAM:

La Block RAM está configurada con un ancho de palabra de 16 bits (1 word) y con un total de 1024 posiciones (1Kword). 500 direcciones almacenan el valor del coseno y otras 500 posiciones almacenan el valor del seno. Aunque los valores del seno y el coseno están generados con 16 bits de precisión, sólo se va a dar auténtico uso a los 11 bits más significativos. El mapa de memoria de la Block RAM es el siguiente:

Dirección (valores decimales)	Datos almacenados
0-499	Tabla del coseno (parte real del factor de rotación). El desfase es de 0º.
500-508	Posiciones inicializadas a 0 (utilizadas por la máquina de estados).
509	Dirección en la que se almacena el coeficiente de escalado del módulo, para

Dirección (valores decimales)	Datos almacenados
	adaptar el valor de la DFT, al valor eficaz al cuadrado (1.49).
510	Media de 5 muestras.
511	Dirección donde se escribe la acumulación de 5 muestras.
512-1011	Tabla del seno (parte imaginaria del factor de rotación). El desfase es de 0º.
1012-1020	Posiciones inicializadas a 0 (utilizadas por la máquina de estados).
1020	Dirección donde se escriben los 16 bits menos significativos para le escalado del módulo.
1021	Dirección donde se escriben los bits más significativos para el escalado del módulo.
1022	Media de 5 muestras.
1023	Coeficiente para el cálculo de la media de 5 muestras (0.2).

Tabla 23: Utilización de la Block RAM

Se ha decidido utilizar una Block RAM de doble puerto. Esto proporciona dos ventajas significativas: aumento de la velocidad de la máquina de estados y facilidad para la utilización del multiplicador implementado. Para garantizar que el sintetizador sólo genere un multiplicador, se ha decido conectar las dos entradas del multiplicador a los dos puertos de salida de la Block RAM (3.2.3). Si se tiene en cuenta que la gran mayoría de operaciones aritméticas realizadas por la máquina de estados, son multiplicaciones, la utilización de la Block RAM en modo doble puerto, supone un aumento de prácticamente el doble, para la velocidad de cómputo.

La Block RAM cuenta con 1024 posiciones de memoria, lo que supone un total de 10 bits para los dos buses de direcciones (doble puerto). Para optimizar la lógica generada, se ha reducido el ancho de los buses de direccionamiento de 9 bits. De tal manera que la Block RAM se ha dividido en 2 zonas independiente de memoria: la RAM A, desde la dirección 0 a la 511 y la RAM B, desde la posición 512 a la 1023. Más adelante se comprobará que supone el ahorro de al menos 1 sumador. Para completar el décimo bit que se debe proporcionar en los buses de direccionamiento, el de la RAM A se ha conectado a ‘0’ y el de la RAM B se conectado a ‘1’ (garantizando así, la independencia de las dos zonas de memoria).

Gestión de las comunicaciones con los bloques locales:

La máquina de estados es el bloque que realiza todas las operaciones aritméticas necesarias, tanto para el cálculo de los puntos de la DFT (media de 5 muestras por el factor de rotación) como para el cálculo del módulo de la DFT a partir de los acumuladores real imaginario.

Como hemos decidido trabajar con un único multiplicador y una única Block RAM, cuando coincidan en el tiempo dos o más peticiones de cálculo de módulo y/o de punto por parte de los bloques locales de DFTs específicas, se debe diseñar una lógica que permita a la máquina de estados calcular los puntos y/o módulos solicitados de una forma secuencial.

Para cubrir la funcionalidad citada se ha diseñado el siguiente circuito:

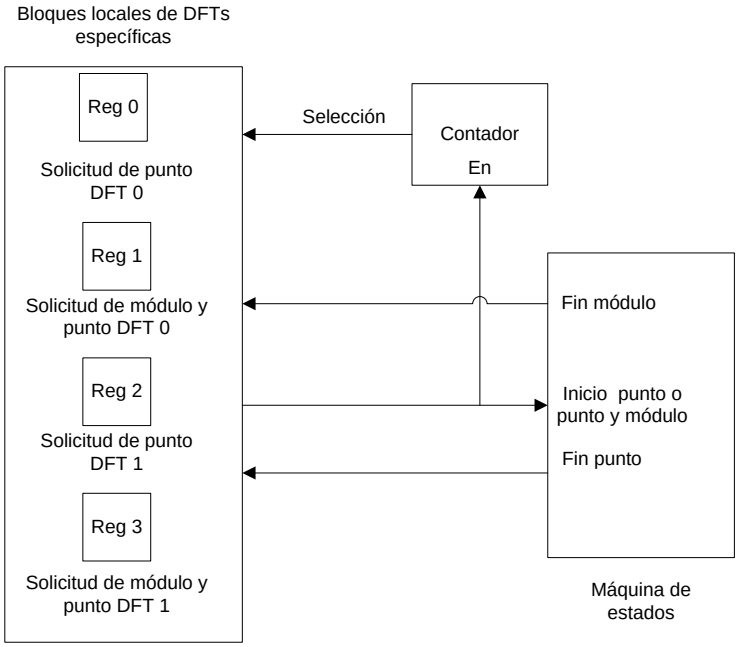

Ilustración 104: Comunicaciones entre el control global y el control local para dos DFTs específicas

Básicamente consiste en un contador que va comprobando canal a canal, la existencia de peticiones de cálculo de punto o punto y módulo por parte de los bloques locales de DFTs específicas. En caso de la existencia de una petición (un '1' en el registro), se iniciará el cómputo correspondiente en la máquina de estados. Hasta que no finalice el cómputo, el registro no tomará nuevamente el valor '0', habilitando de este modo la cuenta del contador. De esta manera, se consigue que todas las operaciones de la máquina de estados sean secuenciales.

Máquina de estados:

Sin más, se expone el diagrama de bloques de la máquina de estados:

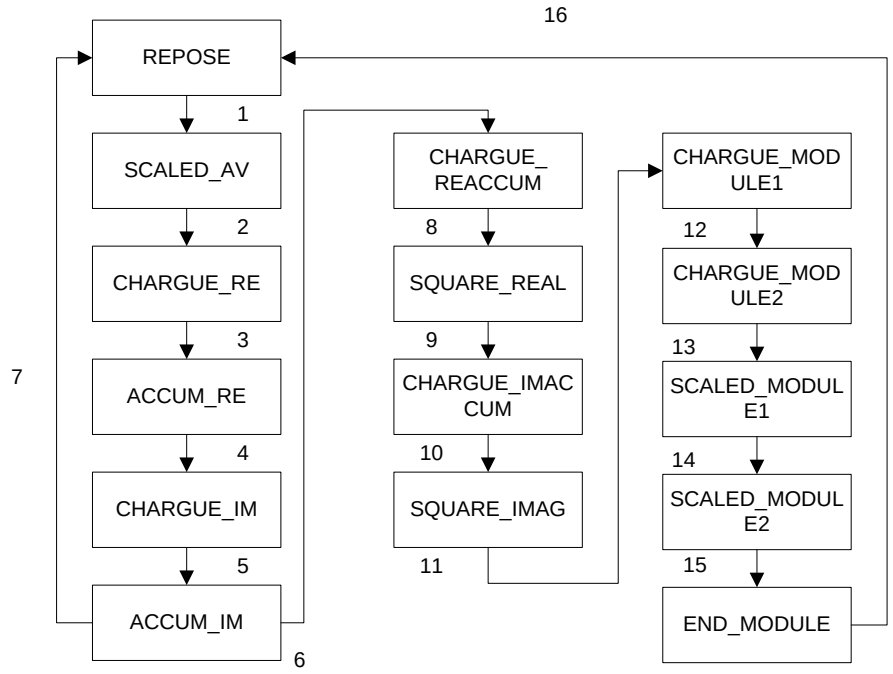

Ilustración 105: Diagrama de estados del control global de la DFT

Descripción de los estados y las transiciones:

1. Partiendo del estado de reposo, ante una petición de cómputo de punto o módulo y punto, se empieza el cálculo de un nuevo punto (SCALED_AV). Se escribe en la dirección 511 de la RAM B, la acumulación previa de 5 muestras. La acumulación proviene del bloque local que ha solicitado el cálculo del punto.

2. De SCALED_AV se pasa al inicio del cálculo de la parte real del punto (CHARGUE_RE). Inicialmente se realiza el cálculo de la media y se multiplica la acumulación de 5 muestras por 0.2 (coeficiente de escalado). Una vez que se ha calculado la media de 5 muestras, se escribe ésta en la dirección 510 de la RAM A y la RAM B. Posteriormente se realiza el producto del resultado de la media por la parte real del factor de rotación (coseno).

3. Al finalizarse el cálculo de la parte real del punto, se pasa a la habilitación de su acumulación (ACCUM_RE) en el acumulador real del bloque local. ACCUM_RE es un estado de espera de un ciclo de reloj (tiempo que tarda en acumularse la parte real del punto).

4. Se pasa de ACCUM_RE a CHARGUE_IM. Finalizado el tiempo de espera, se inicia el cálculo de la parte imaginaria del punto (cargándose en el multiplicador la media de 5 muestras y el seno de la Block RAM).

5. ACCUM_IM es un estado de espera de un ciclo de reloj (tiempo que tarda en acumularse la parte imaginaria del punto). Finalizado el tiempo de espera y si se ha solicitado el cálculo de punto y de módulo, se inicia el cálculo del módulo a partir de los acumuladores real e imaginario del bloque local (se pasa a CHARGUE_REACCUM). Antes de iniciarse el cálculo del módulo, se pone el

acumulador del módulo a 0. En el caso de que no se haya solicitado el cálculo del módulo, al finalizarse el cálculo de la parte imaginaria del punto, se pasa a la habilitación de su acumulación en el acumulador imaginario del bloque local (se pasa a REPOSE). Al mismo tiempo, se emite una habilitación (enable) indicando la finalización del cálculo del punto.

6. Una vez se escribe en la RAM A y en la RAM B los 16 bits más significativos del acumulador real, se pasa al cálculo de su valor al cuadrado pasando a CHARGUE_REACCUM.

7. Finalizado el tiempo de espera y habiéndose solicitado únicamente el cálculo del punto, la máquina de estados vuelve a REPOSE.

8. Finalizado el cálculo de elevar al cuadrado la parte real, se acumula en el acumulador del módulo SQUARE_REAL.

9. Al finalizar el cuadrado de la parte real, se inicia el cálculo de la parte imaginaria al cuadrado (CHARGUE_IMACCUM). Inicialmente se escriben los 16 bits más significativos del acumulador imaginario en la RAM A y en la RAM B.

10. Finalizada la acumulación de la parte imaginaria al cuadrado en el acumulador del módulo, se inicializa la adaptación del módulo obtenido de la DFT específica al valor eficaz al cuadrado de la señal (SQUARE_IMAG).

11. Se inicia la adaptación del módulo al valor de la señal (CHARGUE_MODULE1). Lo primero que se hace es escribir en la RAM B el valor obtenido del módulo. Al ser el ancho de la variable del módulo mayor que el ancho de la RAM, se divide el módulo entre dos posiciones de la RAM. La parte más significativa y la menos significativa.

12. Una vez que se guarda el módulo en la RAM (1 ciclo de reloj), se realiza una multiplicación secuencial con la que se adapta la medida al valor eficaz al cuadrado. La multiplicación se rige por: $V_{RMS}^2 = coef * (Módulo_H * 2^{16} + Módulo_L)$. La constante $coef$ toma el valor de 1.49 (a continuación se explica el escalado).

13. Al finalizar la primera multiplicación se inicia la segunda (1 ciclo de reloj.).

14. Al finalizar la multiplicación secuencial, se indica al módulo local de la DFT específica la finalización del cálculo del módulo.

15. Valor eficaz al cuadrado listo para su lectura. Se pasa a REPOSE, a la espera de una orden de cálculo de punto o punto y módulo.

Medida proporcionada por el módulo de la DFT:

En el apartado 3.2.2 se veía como el valor proporcionado por la DFT específica, adaptado a la señal física medida, guardaba la siguiente relación:

$$V_{picoArmónico} = \frac{|Medida_{DFT}|\ 2}{N}$$

Para $N = 2500$ puntos queda: $V_{picoArmónico} = \frac{|Medida_{DFT}|\ 2}{2500} = \frac{|Medida_{DFT}|}{1250}$

Donde $Medida_{DFT}$, toma el siguiente valor:

$Medida_{DFT} = \sqrt{Acum_{RE}^2 + Acum_{IM}^2}$, por tanto, queda un valor total de:

$$V_{picoArmónico} = \frac{1}{1250}\sqrt{Acum_{RE}^2 + Acum_{IM}^2}$$

La máquina de estados proporciona la siguiente medida:

Los acumuladores real e imaginario presentan un ancho de bits para la parte entera de 24 bits con signo. Para calcular el valor al cuadrado, se toman los 16 bits más significativos, esto es, se dividen los acumuladores entre 2^8 (256). Tomar únicamente los 16 bits más significativos, facilita significativamente, el cálculo del módulo (el ancho de la memoria es de 16 bits). Con lo cual, el valor obtenido es:

$$V_{picoArmónico} = \frac{1}{1250}\sqrt{Acum_{RE}^2 + Acum_{IM}^2} = \frac{1}{4{,}883 * 256}\sqrt{Acum_{RE}^2 + Acum_{IM}^2} =$$

$$= \frac{1}{4{,}883}\sqrt{\frac{Acum_{RE}^2}{256^2} + \frac{Acum_{IM}^2}{256^2}} = \sqrt{\left(\frac{Acum_{RE}^2}{256^2} + \frac{Acum_{IM}^2}{256^2}\right)\frac{1}{4{,}883^2}} =$$

$$= \sqrt{(Medida_{DFT})\frac{1}{4{,}883^2}}$$

Al valor obtenido se le aplica un coeficiente de adaptación para obtener el valor eficaz al cuadrado del harmónico:

$V_{rms} = \frac{V_{picoArmónico}}{\sqrt{2}} = \frac{1}{\sqrt{2}}\sqrt{(Medida_{DFT})\frac{1}{4{,}883^2}}$, el valor eficaz al cuadrado será: $V_{rms}^2 = (Medida_{DFT})\frac{1}{2\ x\ 4{,}883^2} = (Medida_{DFT})\frac{1}{1{,}490\ x\ 2^5}$, de aquí se deduce el coeficiente de escalado 1,490 utilizado en los apartados anteriores.

3.4 Validación del módulo de la DFT

En este apartado se va a verificar el funcionamiento del módulo de la DFT para señales reales. La idea es utilizar un generador de señales convencional para poder variar la señal de entrada al módulo de la DFT de una manera rápida y sencilla. Para ello, se va a implementar el diseño sobre el kit de Xilinx Spartan-3A/3AN FPGA Starter Kit (FPGA Spartan3-700AN). La ventaja de utilizar el kit, es que se dispone de dos periféricos fundamentales para trabajar con el generador de señales: 2 ADCs y 4 DACs.

Ya que el generador de señales sólo dispone de un único canal, lo que se pretende es generar una señal de múltiples armónicos (señal cuadrada, triangular, etc). Utilizándose 2 de los 4 DACs disponibles, se espera poder representar 2 de los armónicos a partir de la señal de entrada (generando para ello dos bloques locales de DFTs específicas).

A continuación se expone el diagrama de bloques empleado en la implementación:

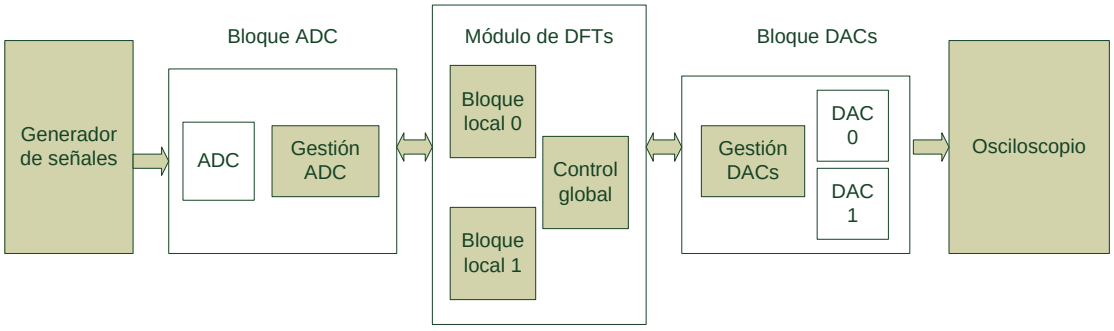

Ilustración 106: Diagrama de bloques para la validación de la DFT

De cara a la simulación se ha diseñado un modelo del ADC y un emulador para el generador de señales. Esto hará de la simulación una herramienta más eficaz para la detección de errores en el sistema. La creación de los modelos para la simulación se describe en el apartado 3.4.2.

A continuación se describen los bloques para la implementación.

3.4.1 Gestión del reset y del reloj

El módulo de la DFT no está diseñado para funcionar a una frecuencia específica en concreto, aunque cuando se integre en la plataforma de control del convertidor, su frecuencia en servicio será de 20 MHz. Por este motivo, se ha decidido validar el diseño para una frecuencia de 20 MHz.

El kit de Xilinx dispone de un oscilador de 50 MHz (el mismo oscilador empleado en el módulo de muestreo y gestión de datos en memoria). La idea es utilizar los DCMs disponibles en la FPGA para la gestión del reset y del reloj. Como se ha comentado anteriormente, a partir del reloj de 50 MHz se va a generar el reloj de 20 MHz. El módulo de la DFT es síncrono con el flanco de subida del reloj mientras que los

circuitos de control del ADC y de los DACs son síncronos con el flanco de bajada y de subida del reloj. Para simplificar la gestión del reloj y evitar caer en una lógica de sincronismo complicada, se ha decidido implementar el siguiente circuito:

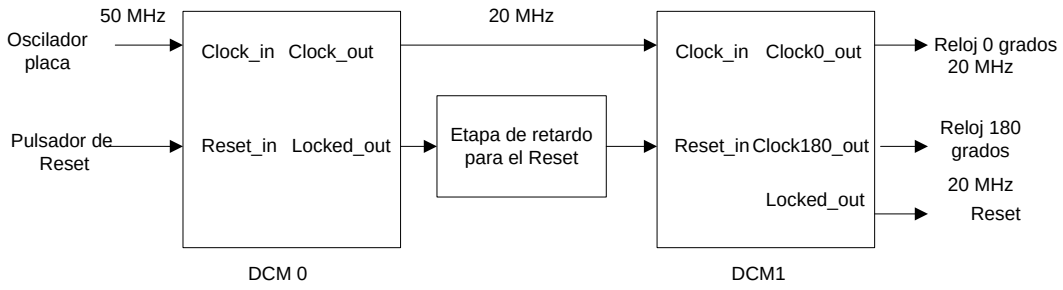

Ilustración 107: Gestión del reset y del reloj para la implementación del módulo DFT

Al emplear dos relojes de 20 MHz desfasados entre sí 180º, se consiguen sincronizar el flanco de bajada y el flanco de subida en el mismo instante de tiempo. Las entidades síncronas con el flanco de subida del reloj emplearán el reloj sin desfase, las entidades síncronas con el flanco de bajada del reloj, emplearán el reloj con el desfase de 180º. Todos los biestables serán síncronos con el reset generado por el segundo DCM.

3.4.2 Uso de ADC y DACs para tareas de adquisición y transmisión de datos.

Control del ADC y gestión de las muestras de entrada:

Se dispone de dos ADCs de Digilent (ADCS7476) de 12 bits con comunicación serie (protocolo SPI) en el mismo encapsulado. Al disponer de una única sonda en el generador de señales, se va a hacer uso de un único ADC.

Se ha diseñado un circuito que sea capaz de implementar una comunicación SPI de 12 bits. En realidad, en cada comunicación SPI se reciben 16 bits del ADC, siendo los 4 primeros bits completamente desechables. Se debe tener en cuenta que el control del ADC es síncrono con el flanco de bajada del reloj. Además, la comunicación SPI puede funcionar con una frecuencia máxima de 20 MHz. Por lo que se va a emplear el reloj de 20 MHz generado con los DCMs. El rango de la amplitud de entrada al ADC se comprende entre 0 y 3,3 voltios. La medida siempre va a ser positiva, por lo que en la gestión de las muestras de entrada, se le va a añadir el bit de signo a '0'. Teniéndose el total de 13 bits solicitados por el módulo de la DFT.

Para la gestión de las muestras de entrada se ha diseñado un temporizador. El temporizador genera una habilitación (enable) cada 8 microsegundos, indicándole así al módulo de la DFT la existencia de nuevas muestras. Siempre que el ADC realice una conversión, la muestra se almacena en un bloque de registros. El esquema del circuito es el siguiente:

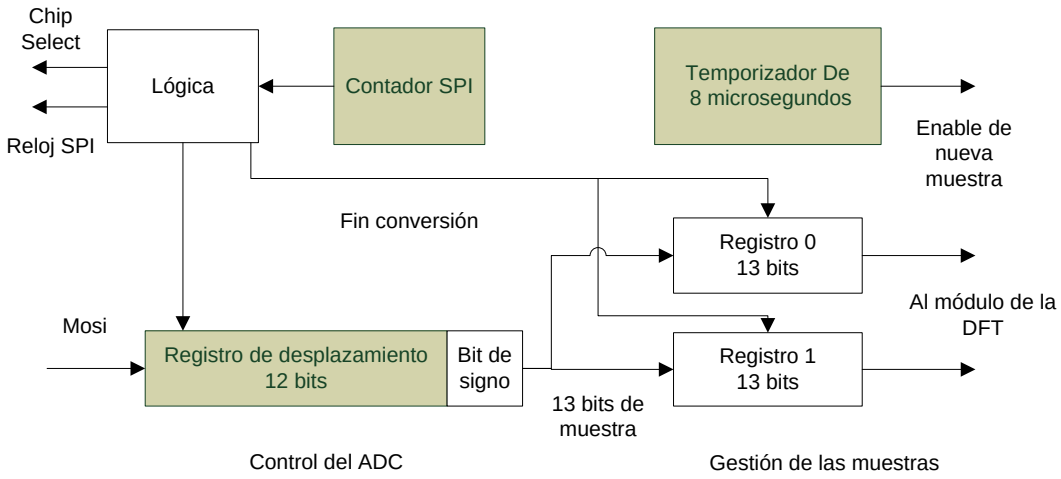

Ilustración 108: Control del ADC y gestión de las muestras de entrada

Para cualquier duda sobre el funcionamiento del ADC, consultar la hoja de características del "ADCS7476" de Digilent.

Control del los DACs y gestión de los datos de salida:

Se van a utilizar 2 de los 4 DACs presentes en la placa. Cada DAC va a estar asociado a un armónico de la señal muestreada con el ADC, el DAC 0 al tercer armónico (150 Hz) y el DAC 1 al primer armónico (50 Hz). El protocolo de comunicaciones para los DACs es, nuevamente, el SPI. Cada conversión del DAC consta de 24 bits:

$$COMMAND(4\ bits) + \ ADDRESS(4\ bits) + data(16\ bits)$$

Command hace referencia a la operación que se va a realizar, en este caso siempre va a tomar el valor de x"3" (write to and update address). Por address se entiende el DAC en el que se quiere escribir, que puede tomar el valor de x"0" (DAC 0) o x"1" (DAC 1), dependiendo del armónico que se haya medido. Data toma el valor de los 16 bits más significativos del módulo medido por la DFT.

La gestión de los datos de salida consta de dos partes: 2 registros que almacenan los 24 bits solicitados por el DAC cada vez que finaliza el cálculo del respectivo armónico y un circuito para la gestión de peticiones de conversión al control de los DACs.

El valor esperado al medir con el osciloscopio es el siguiente:

$V_{rms}^2 = (Medida_{DFT})\frac{1}{1{,}49\ x\ 2^5}$. El DAC desprecia los 6 bits menos significativos de los 25 bits con los que se representa V_{rms}^2, con lo que queda:

$$Salida_{DAC} = \frac{Medida_{DFT}}{2^{(6)}} = \frac{Medida_{DFT}}{64}$$

Con lo que quedarían un total de 19 bits con los que se representa el valor eficaz. Como el DAC solo puede tomar palabras de 16 bits, se van a generar señales de

entrada para las que la medida de la DFT específica no requiera de los 3 bits más significativos.

La salida de los DACS varía en un rango comprendido entre 0 y 3,3 voltios. No toman valores negativos, lo que no supone ningún problema. Debido a que el módulo de la DFT, proporciona únicamente, números positivos.

El funcionamiento de los DACs se puede consultar en el documento "LTC2604, Quad 16-Bit Rail-to-Rail DACs in 16-Lead SSOP" de Linear Technology.

3.4.3 Modelado del generador de señales y del ADC para la simulación

Emulador para el generador de señales:

Se ha diseñado un fichero en VHDL con el objeto de comprobar el comportamiento del módulo de la DFT ante múltiples señales en un entorno de simulación. Es un fichero no sintetizable ya que se hace uso de señales tipo real y de funciones no sintetizables para la generación del seno. Se ha decidido no hacerlo sintetizable por la facilidad asociada a la utilización del generador de señales.

El emulador es capaz de generar señales de un modo parecido al generador: Cuadradas, senoidales y triangulares con una frecuencia, amplitud, resolución y nivel de continua completamente parametrizables.

En general, la señal más interesante a la que aplicar la DFT, es una señal cuadrada de amplitud máxima que esté en fase con la parte real o imaginaria del vector de rotación. Esto somete a todos los acumuladores y multiplicadores presentes en el módulo a trabajar a su capacidad máxima. Si los acumuladores no se saturan en estas condiciones, no se van a saturar con ningún otro tipo de señal. Además, las señales cuadradas presentan infinitos armónicos, por lo que pueden ser útiles para comprobar la resolución proporcionada por el módulo en la medida del 3º o 5º armónico.

Si se desea simular el módulo de la DFT por sí solo, el emulador genera, cada 8 microsegundos, la habilitación (enable) de nueva muestra que en condiciones normales de funcionamiento sería generado por el control de la FPGA.

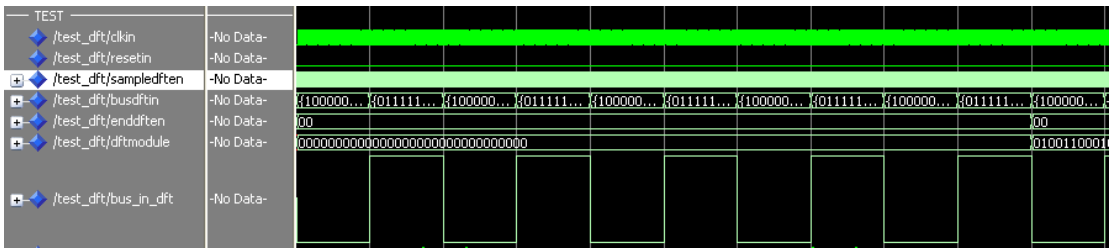

Ilustración 109: Onda cuadrada generada por el emulador

Se recomienda el uso del software Modelsim (versión 6.5C) para la simulación del módulo de la DFT. La ventaja de Modelsim sobre la versión libre de ISIM, es la posibilidad de visualizar las señales en formato analógico, algo imposible en la versión libre de ISIM. Es difícil o muy difícil la tarea de depuración de fallos o comprensión del funcionamiento de un viendo únicamente señales digitales. Del mismo modo, resulta interesante comprobar que la inicialización del seno y del coseno en la Block RAM está bien implementada, así como las formas de onda de los acumuladores real e imaginario.

Una vez se ha descrito el bloque empleado para la generación de señales en la simulación, se describen los bloques para la implementación de la Ilustración 106.

Modelo del ADC:

Para la simulación se ha generado un modelo del ADC. Básicamente consiste en un registro de desplazamiento que registra los datos producidos por el emulador del generador de señales (solo cuando el chip select y el reloj SPI toman el valor válido).

Resultados de la simulación:

Mediante simulación se ha verificado el funcionamiento del módulo de la DFT. Concretamente se ha comprobado que la lógica diseñada para el control local y global del módulo de la DFT funciona correctamente. También se ha simulado teniendo en cuenta las condiciones más desfavorables, de cara a la saturación del multiplicador y los acumuladores. Dichas condiciones consisten en una señal cuadrada de amplitud máxima en fase con la parte real o imaginaria del factor de rotación.

Una vez se ha comprobado el funcionamiento del diseño mediante simulación, se procede a la implementación del circuito sobre una FPGA. A continuación se describen las medidas realizadas para la validación del diseño.

3.4.4 Aplicación de la DFT a señales empleando un generador de funciones

Se va a someter al módulo de la DFT a una señal cuadrada de 50 Hz. Los DACs 0 y 1 van a proporcionar la amplitud del tercer y primer armónico respectivamente.

Mediciones esperadas:

Antes de realizar las medidas con el osciloscopio, se ha hecho un estudio teórico. Con este estudio se pretende razonar el valor esperado en las medidas.

Medidas físicas:

El canal 2 se corresponde con el valor para el primer armónico (50 Hz), el canal 3 proporciona la medida para el tercer armónico (150 Hz) y el canal 1 se corresponde con la señal sobre la que se aplica la DFT específica.

La nomenclatura empleada en los cálculos expuestos a continuación es la siguiente:

- $MedidaOsciloscopio_0$ se corresponde con el tercer armónico de la señal, dicha medida se obtiene del DAC 0.
- $MedidaOsciloscopio_1$ se corresponde con el primer armónico de la señal, dicha medida se obtiene del DAC 1.

Primera medida: Onda Cuadrada de 50 Hz, 1,5 Vp y 1,5 Vcc:

Señal de entrada: onda cuadrada de 50 Hz con una amplitud de 1,5 voltios pico y un nivel de continua de 1,5 voltios (solo toma valores positivos y no supera los 3,3 voltios del ADC).

$Armónico_1 = \frac{4}{\pi} * 1{,}5 = 1{,}91\, Vpico$, la medida digitalizada queda (escala decimal): $ADC = 2^{12} * \frac{1{,}91}{3{,}3}\frac{1}{\sqrt{2}} = 1676\, Valor_{rms}$

$Armónico_3 = \frac{4}{\pi} * \frac{1{,}5}{3} = 0{,}636\, Vpico$, la medida digitalizada queda (escala decimal): $ADC = 2^{12} * \frac{0{,}636}{3{,}3}\frac{1}{\sqrt{2}} = 559\, Valor_{rms}$

Estos son los valores eficaces al cuadrado teóricos para el armónico 1 y 3 de la señal cuadrada. La medida proporcionada por el módulo de la DFT es:

$$V_{rms}^2 = (Medida_{DFT)}\frac{1}{1{,}49\ x\ 2^5}$$

$$Medida_1 = 1676^2 = 2\,808976\ V_{rms}^2$$

$$Medida_3 = 559^2 = 312\,481\ \ V_{rms}^2$$

Ésta es la medida digital (escala decimal) proporcionada por el módulo de la DFT para los armónicos 1 y 3. Se recuerda que la medida es una señal con el siguiente formato de bits [24 : 0]. En el DAC únicamente se toma el valor [22 : 6], con lo cual tenemos a la entrada de los DACs los siguientes números, de nuevo, en escala decimal:

$$DAC_0 = \frac{312\,481}{2^6} = 4882{,}5$$

$$DAC_1 = \frac{2\,808976}{2^6} = 43890{,}3$$

Por último, se calcula el valor a la salida de los DACs, o lo que es lo mismo, el valor que se espera medir con el osciloscopio:

$MedidaOsciloscopio_0 = \frac{4882{,}5}{2^{16}} * 3{,}3 = 0{,}246\, Vcc$, para el tercer armónico de la onda cuadrada.

$MedidaOsciloscopio_1 = \frac{43890,3}{2^{16}} * 3,3 = 2,2\ Vcc$, para el primer armónico de la onda cuadrada.

El valor que se va a medir en el osciloscopio para cualquier armónico, se deduce de la siguiente ecuación:

$$MedidaOsiloscopio = Varmónico^2_{pico} * \frac{2}{3,3} \text{ (medido en } Vcc)$$

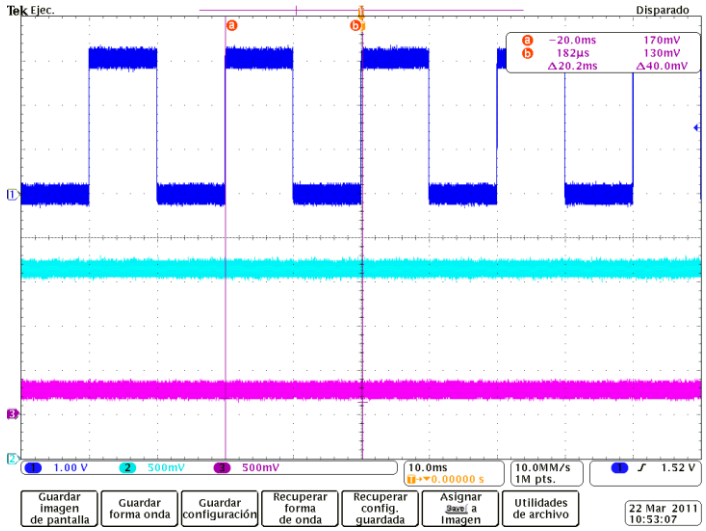

Ilustración 110: DFT de onda cuadrada de 50 Hz, 1,5 Vp y 1,5 Vcc

Para el armónico fundamental se miden 2,2 Vcc. Para el tercer armónico se miden 250 mVcc. Estos valores coinciden plenamente con obtenidos en los cálculos teóricos.

Segunda medida: Onda cuadrada de 45 Hz, 1,5 Vp y 1,5 Vcc:

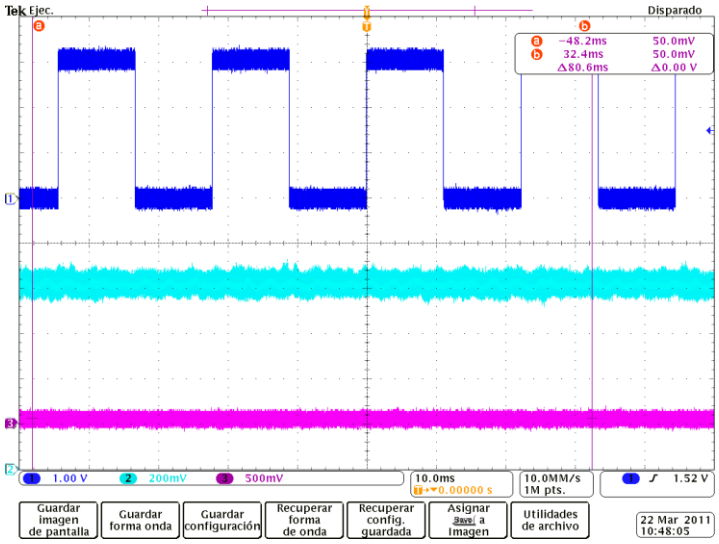

Ilustración 111: DFT de onda cuadrada de 45 Hz, 1,5 Vp y 1,5 Vcc

Se comprueba como varía la medida de la DFT ante variaciones de frecuencia en la señal de entrada. El valor de pico para el primer armónico ha pasado de valer 2,2 Vcc a

tomar el valor de 0,7 Vcc.La componente se reduce a un 36% del valor obtenido anteriormente.

Tercera medida: Onda senoidal de 50 Hz, 1,5 Vp y 1,5 Vcc:

Empleando la relación calculada anteriormente se obtiene el valor teórico esperado:

$$MedidaOsiloscopio = Varmónico^2_{pico} * \frac{2}{3{,}3} \text{ (medido en } Vcc)$$

$MedidaOsciloscopio_0 = 0^2 * \frac{2}{3{,}3} = 0{,}000\, Vcc$, para el tercer armónico de la onda senoidal (se supone una THD igual a 0 para el generador de funciones).

$MedidaOsciloscopio_1 = 1{,}5^2 * \frac{2}{3{,}3} = 1{,}364\, Vcc$, para el primer armónico de la onda senoidal.

Midiendo con el osciloscopio (Ilustración 112) se comprueban los cálculos teóricos. Una onda senoidal pobre en armónicos (THD próxima a 0), da como resultado, un tercer armónico casi inexistente (prácticamente 0 Vcc) y una medida para el primer armónico de 1,4 Vcc. Ambos valores concuerdan con los cálculos teóricos.

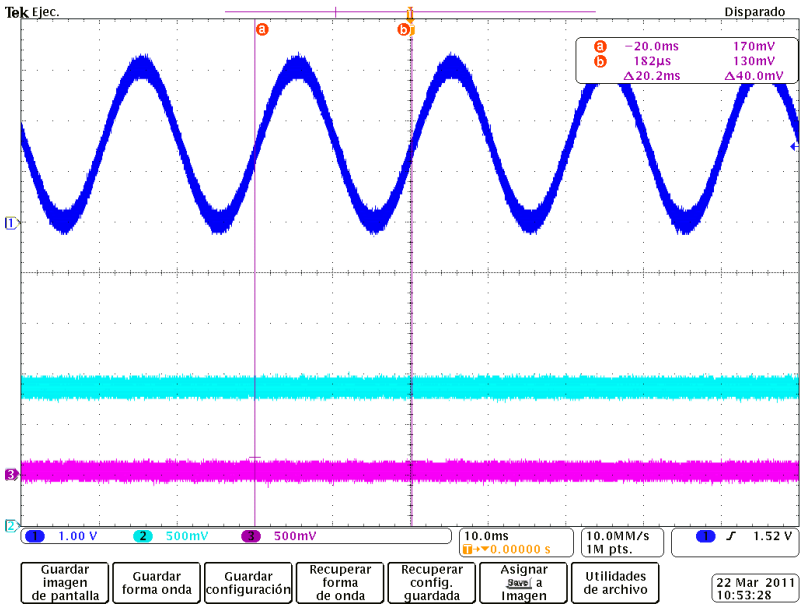

Ilustración 112: DFT de onda senoidal de 50 Hz, 1,5 Vp y 1,5 Vcc.

Cuarta medida: Onda triangular de 50 Hz, 1,5 Vp y 1,5 Vcc.

Primero se calcula el valor teórico para el primer y tercer armónicos.

$$MedidaOsiloscopio = Varmónico^2_{pico} * \frac{2}{3{,}3} \text{ (medido en } Vcc)$$

$MedidaOsciloscopio_0 = 0^2 * \frac{2}{3{,}3} = 0{,}000\,Vcc$, para el tercer armónico de la onda triangular (se supone una THD igual a 0 para el generador de funciones).

$MedidaOsciloscopio_1 = (1{,}5 * \frac{8}{\pi^2})^2 * \frac{2}{3{,}3} = 0{,}896\,Vcc$, para el primer armónico de la onda triangular.

Se puede ver en la Ilustración 113 como las medidas concuerdan con los cálculos teóricos. 0 Vcc para el tercer armónico y 0,9 Vcc para el primer armónico.

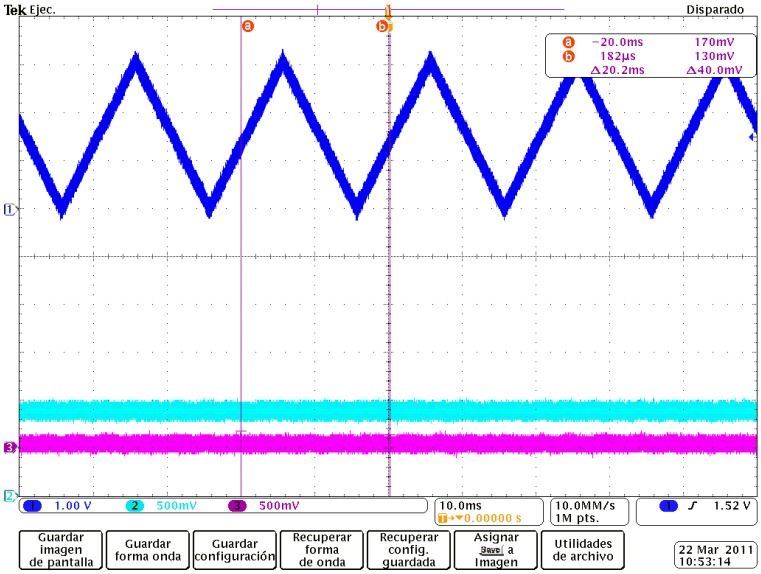

Ilustración 113: DFT de onda triangular de 50 Hz, 1,5 Vp y 1,5 Vcc

3.5 Resultados de síntesis

En este apartado se van a proporcionar el área y la frecuencia máxima de funcionamiento tras sintetizar el módulo de la DFT:

La FPGA presente en el diseño es la Spartan3–1600E. Dicha FPGA se monta sobre el "Spartan-3E FPGA Industrial Micromodule" de Trenz. En este caso, se ha configurado el circuito para que calcule uno de los siguientes armónicos para 5 señales diferentes: 5, 4, 3, 2, 1 (señal 1 armónico 1, señal 2 armónico 2, etc.).

Área:

Los resultados de área se obtienen del MAP con el software ISE.

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	584	29,504	1%	
Number of 4 input LUTs	1,018	29,504	3%	
Number of occupied Slices	578	14,752	3%	
Number of Slices containing only related logic	578	578	100%	
Number of Slices containing unrelated logic	0	578	0%	
Total Number of 4 input LUTs	1,100	29,504	3%	
Number used as logic	1,018			
Number used as a route-thru	82			
Number of bonded IOBs	102	250	40%	

Tabla 24: Área módulo de la DFT

Frecuencia máxima:

Los resultados de frecuencia se obtienen del Place&Route con el software ISE. La frecuencia de funcionamiento en servicio va a ser de 20 MHz. Por tanto, se cuenta con un holgado margen para un posible aumento de la frecuencia de funcionamiento.

```
Design statistics:
   Minimum period:  21.624ns(1)    (Maximum frequency:  46.245MHz)
```

Ilustración 114: Frecuencia máxima módulo de la DFT

Se observa que el diseño cumple con las especificaciones de tiempo.

4 Presupuesto

El presupuesto se deriva en tres partes fundamentales: los costes amortizables, presupuesto de material y el presupuesto de personal:

Costes amortizables (Instrumentación):

Instrumentación	Coste	Período de amortización	Tiempo de utilización	Coste de utilización
Ordenador	800 €	5 años	100 días	44,44 €
Osciloscopio	12000 €	10 años	100 días	333,33 €
Licencia Matlab	3250 €	5 años	100 días	180,55 €
Licencia Modelsim	Versión libre	-	-	-
Liencia ISE	Versión libre	-	-	-
Generador de funciones	1500 €	10 años	100 días	41,66 €
Total Costes amortizables	-	-	-	599,98 €

Tabla 25: Costes amortizables

Presupuesto de material:

Material	Modelo	Coste unitario	Medición	Coste total
"Spartan-3E FPGA Industrial Micromodule"	TE0300-01IB FPGA: XC3S1600E 4FGG320 4I	201,11 €	1	201,11 €
Prototyping Carrier Board for Industrial Micromodule	TE0303-01	46,41 €	1	46,41 €
Spartan-3 FPGA Starter Kit	Spartan-3AN (XC3S700AN-FG484)	140,3 €	1	140,3 €
Pmod AD1 (convertidor analógico/digital)	23329	17,35 €	1	17,35 €
Presupuesto total Material	-	-	-	405,17 €

Tabla 26: Presupuesto de material

Presupuesto de personal:

Tarea	**Profesional**	**Coste/ hora**	**Número de horas**	**Coste total**
Diseño y desarrollo del proyecto	Ingeniero técnico industrial: Electrónica industrial	13,7 €	500	6850 €
Presupuesto total personal	-	-	-	6850 €

Tabla 27: Presupuesto de personal

El presupuesto total se obtiene de la suma de los costes amortizables, el presupuesto de personal y el presupuesto de material: $599{,}98\ € + 405{,}17\ € + 6850\ € = \mathbf{7855{,}15}\ \boldsymbol{EUROS}$.

5 Conclusiones y líneas de investigación futuras

En este apartado se va ha hacer una comparativa entre los objetivos iniciales del proyecto y los alcanzados a la conclusión de éste. Finalmente se mencionarán las posibles aplicaciones y/o ampliaciones más interesantes.

Inicialmente se propuso la creación de dos módulos VHDL para la plataforma de control del convertidor. Una traza de averías (módulo de muestreo y gestión de datos en memoria) y el algoritmo de la DFT centrado en un armónico (módulo de la DFT).

Traza de averías:

Objetivos iniciales: La traza de averías debía de ser capaz de procesar las muestras de un máximo de 4 tarjetas I/O y 2 registros de estado, proporcionarlas un formato adecuado para su clasificación y guardarlas en una memoria DDR funcionando a modo de LIFO. Al producirse una avería en el convertidor, el control de la FPGA procedería a la lectura de ésta.

Objetivos alcanzados: Se puede concluir que se han cumplido satisfactoriamente todos los objetivos iniciales del proyecto. Se ha conseguido diseñar una traza de averías con una capacidad de gestión máxima de tarjetas I/O y registros de estado 6 veces superior a la requerida. Se han añadido circuitos para verificar y validar el correcto funcionamiento del módulo, como por ejemplo, comprobación de la pérdida de muestras en el multiplexor al interfaz y en el circuito de gestión de muestras. Además se han diseñado dos tests para la memoria DDR, uno que verifica y borra todas las posiciones de memoria y otro que únicamente borra la memoria.

Líneas de investigación futuras:

- Diseño de una lógica que permita el aumento de la velocidad de lectura, en el caso de que en un futuro, aumente la velocidad de procesamiento de datos por parte del control de la FPGA.
- Diseño de una lógica adicional que posibilite que la memoria DDR pueda ser utilizada por otros módulos de la plataforma de control.

DFT:

Objetivos iniciales: se requería diseñar un módulo capaz de calcular la DFT centrada en un armónico para un número programable de señales y/o armónicos. La medida debía proporcionarse en notación polar (despreciándose el argumento).

Objetivos alcanzados: Se ha conseguido cumplir con todos los requisitos iniciales. Adicionalmente se ha añadido una lógica que proporciona como medida el valor eficaz del armónico al cuadrado.

Líneas de investigación futuras:

- Cálculo de la THD. Es muy habitual encontrar inversores que incluyen el cálculo de la THD como medida de la calidad de la señal que están generando. Tomando como referencia el módulo de la DFT se puede diseñar un circuito para el cálculo de la THD.

6 Bibliografía

[1] S.A. Pérez, E. Soto, S. Fernández, "Diseño de sistemas digitales con VHDL", Thomson, 1ª Edic., 2002.

[2] J.G. Proakis, D.G. Manolakis, "Tratamiento digital de señales", Prentice Hall, 3ª Edic., 1998.

[3] Steven T. Karris, "Signals and Systems with MATLAB Computing and Simulink Modeling", Orchard Publications, 3ª Edic., 2007.

[4] Steven T. Karris, "Introduction to Simulink with Engineering Applications", Orchard Publications, 1ª Edic., 2006.

[5] UG334, "Spartan-3A/3AN FPGA Starter Kit Board User Guide", Xilinx, V1.1, 2008.

[6] UG331, "Spartan-3 Generation FPGA User Guide", Data sheet, Xilinx, V1.6, 2009.

[7] DS312, "Spartan-3E FPGA Family", Data Sheet, Xilinx, V3.8, 2009.

[8] UG086, "Memory Interface Solutions User Guide", Xilinx, V3.5, 2010.

[9] DS615, "Discrete Fourier Transform", Xilinx, V3.1, 2009.

[10] "512MbDDR2", Data sheet, Micron Technology, 2004.

[11] "512MBDDRx4x8x16", Data sheet, Micron Technology, 2000.

[12] "ADCS7476", Data sheet, National Semiconductor, 2010.

[13] "LTC2624 Quad DAC", Data sheet, Linear Technology, 2004.

[14] "Simulink Fixed Point User`s Guide", MathWorks, 2010.

[15] http://www.xilinx.com/. Página web de Xilinx.

[16] http://www.mathworks.com/products/matlab/. Página web de MATLAB.

[17] http://es.wikipedia.org/wiki/Wikipedia:Portada. Página web de Wikipedia.